Chapter

6

Visual Basic: While Loop

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

- **Understanding a Fibonacci Sequence**
- **Using a Loop with the Do While Function**
- **Opening Visual Basic Editor**
- **Beginning a New Visual Basic Project**
- **Laying Out a User Input Form in Visual Basic**
- **Insert a Label into a Form**
- **Insert a Textbox into a Form**
- **Inserting more Labels and Textboxes into a Form**
- **Using a Read Only Textbox with a Vertical Scroll Bar**
- **Insert Command Buttons into a Form**
- **Adding a Copyright Statement to a Form**
- **Adding Comments in Visual Basic to Communicate the Copyright**
- **Declaring Variables in a Program with the Dimension Statement**
- **Setting Variables in a Program**
- **Processing Inside of a Loop**
- **Resetting the Data**
- **Exiting the Program**
- **Running the Program**

Understanding a Fibonacci Sequences

In this program, we will create a form and ask for a starting number and the second number in a Fibonacci Sequence. We will also ask for the quantity of numbers in the set. When we calculate the set, we will use a while loop to compute each successive number in the series after the first two, so this exercise is an excellent opportunity to learn how to execute a while loop. The answer will be shown in a read only textbox that has a vertical scroll bar.

A Fibonacci Sequence takes the first two numbers (number1 and number2) and adds them together to make the subsequent number (number3). We write the new number (number3) to the list. In example, we take 0 and 1 and add them together and we calculate 1. Our new set of numbers is {0,1,1}. We set the value of number2 to the variable number1 and the value of number3 to the variable number2. We add number1 to number2 again and assign the quantity to the variable number3. In our example, we take 1 and 1 and add them together and we compute 2. Our new set is {0,1,1,2}. We continue the process and the sequence looks like the following set.

{0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610}

As in previous chapters, we will learn new techniques in showing information on a form and to code the program, so we expect our students to enjoy this chapter.

Using a Loop with the Do 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.

With 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. As with engineering students, we want to find a problem that needs to cycle through a repetitive procedure that they can use. The math students often have to create Fibonacci sequences, so here is a good program to help them do the math. In our while loop in Visual Basic, we are going to add two numbers and concatenate the new number to a string. 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 Basic, 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 amount of numbers in the Fibonacci sequence because we ask for the quantity on the form. The test is simple. We will stay in the loop as long as the variable counter is less than quantity.

If the quantity is 10 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.

Open the Visual Basic 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 File on the Menu Bar and New Project.

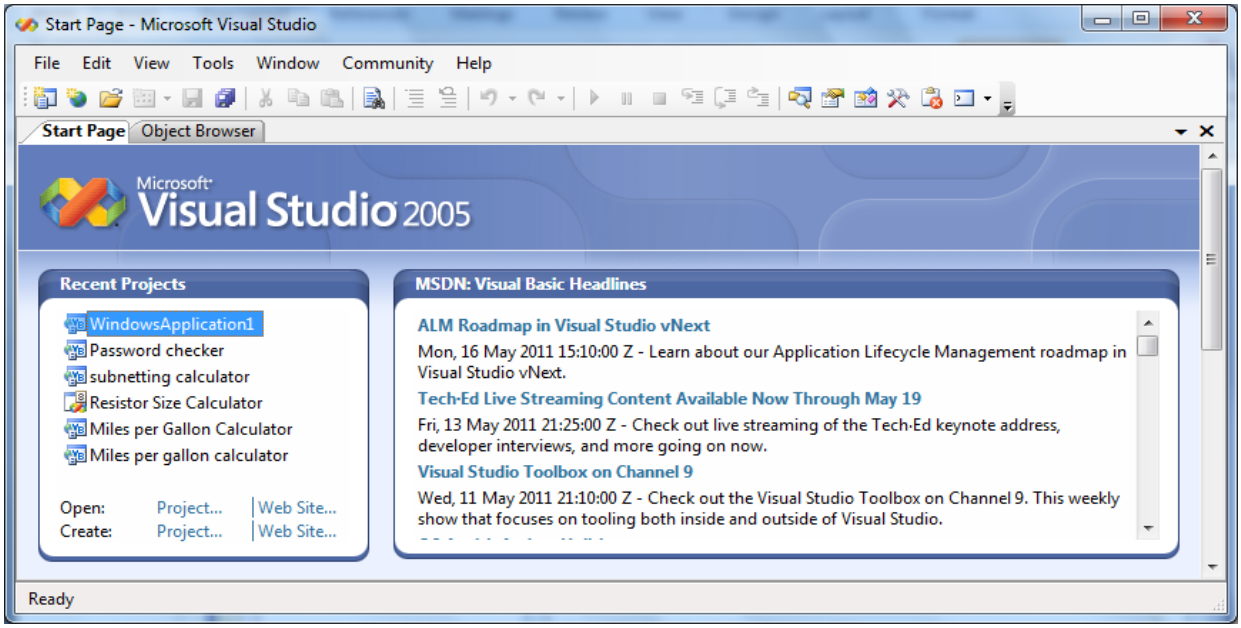

Figure 6.1 – The Start Page

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

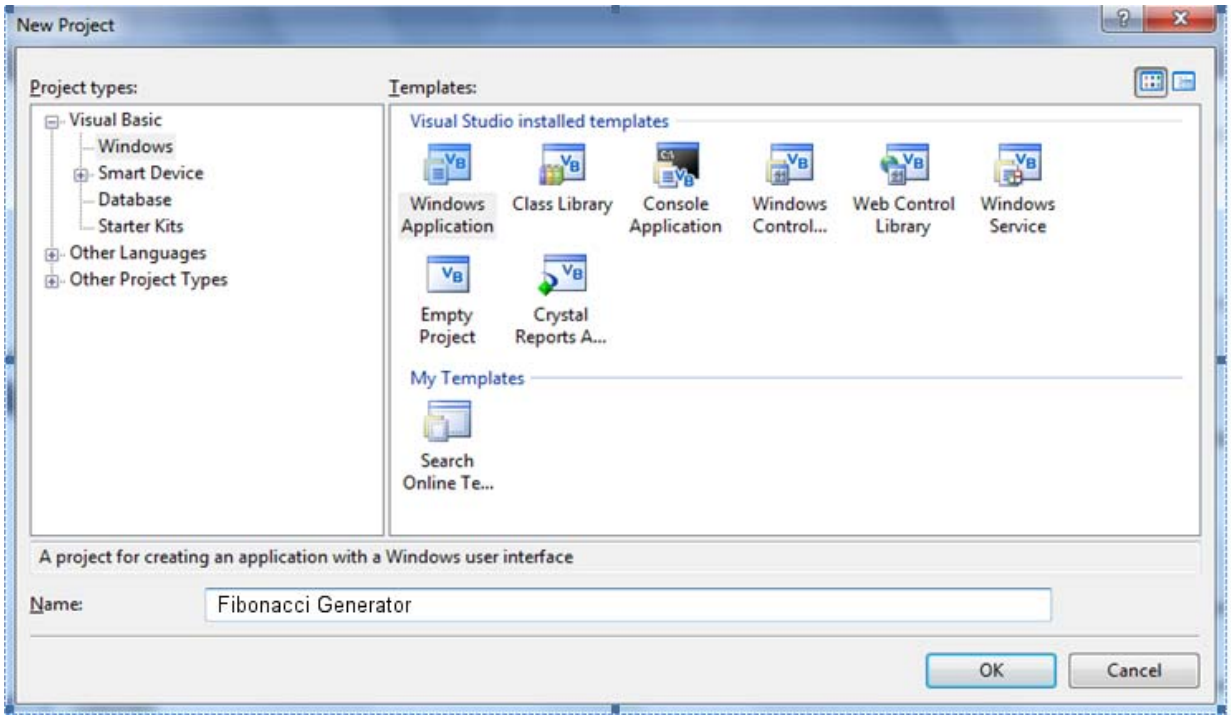

Figure 6.2 – New Project

With the Visual Basic Editor open, select **File** on the Menu Bar and select **Save All**. For the location, we will browse to the folder "Visual Basic Projects" that we made in Chapter 2. We will name this project "Fibonacci Generator". A folder called "Fibonacci Generator" will be made and all the files for the program will be located in the folder.

Beginning a New Visual Basic 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, Fibonacci Generator. We will have a label for the input textboxes. We will have a textbox to key in the first two numbers and the amount of numbers in the sequence. We will have three command buttons, Calculate, 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.

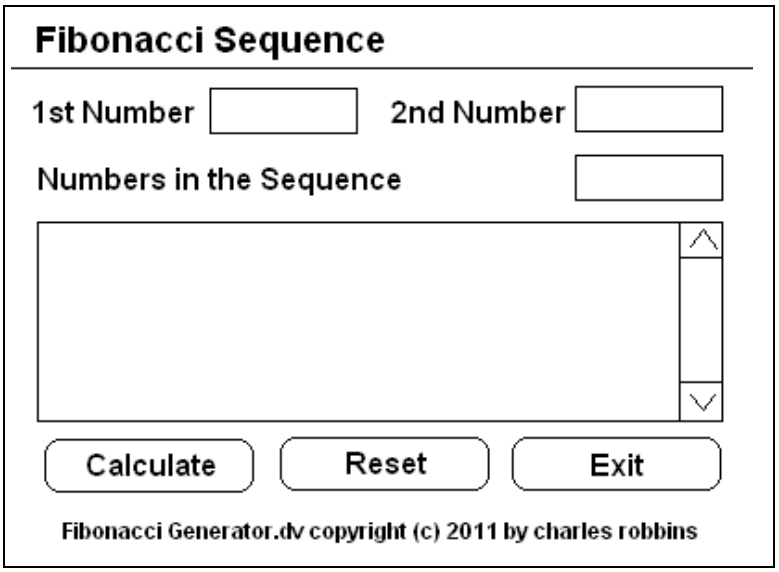

Figure 6.3 – Sketch of the Fibonacci Generator Form

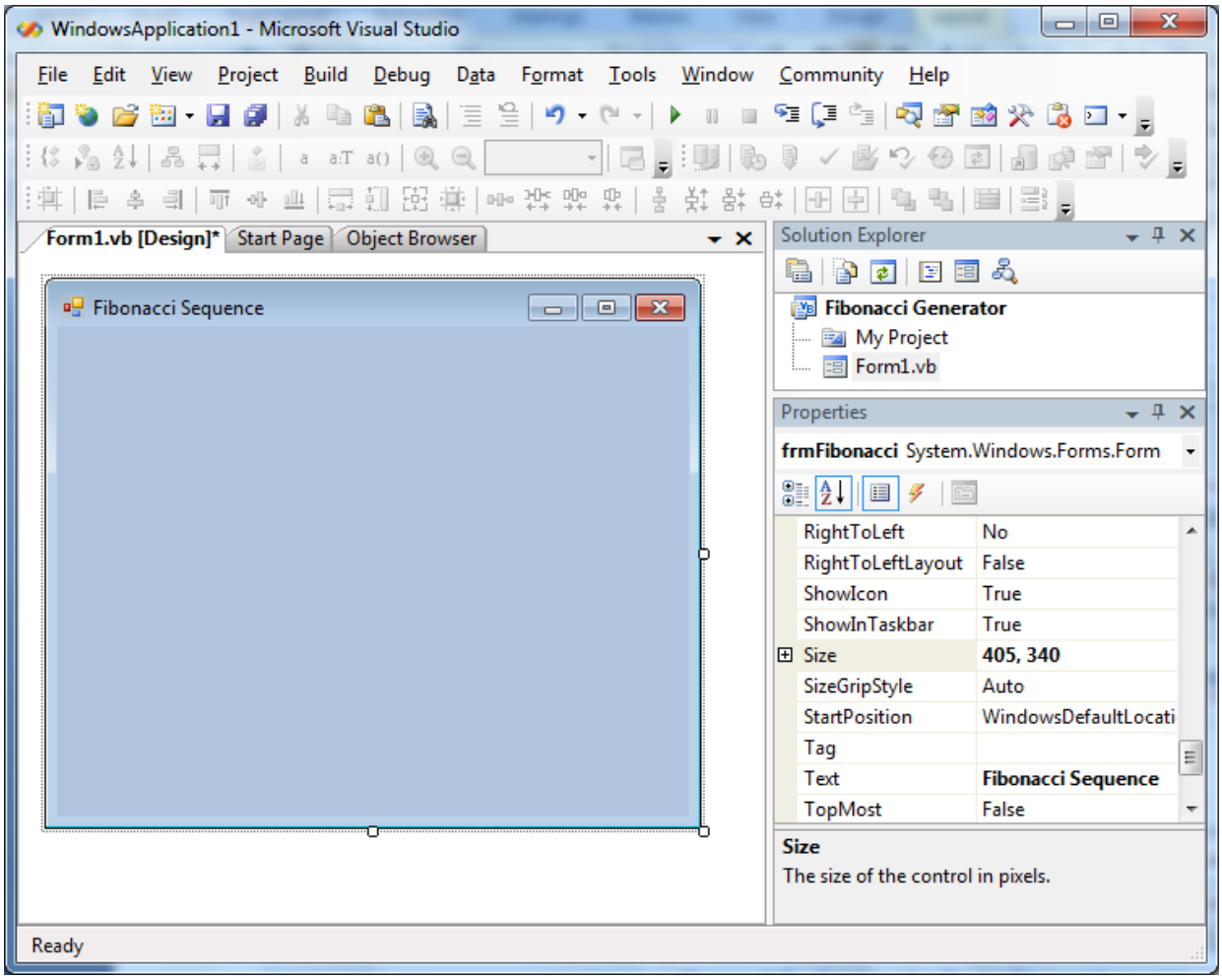

Figure 6.4 – Designing the Fibonacci Generator Form in Visual Basic

Laying Out a User Input Form in Visual Basic

We will change the **Text** in the Properties pane to Fibonacci Generator to agree with the sketch in Figure 6.3. Go ahead and change the form in two other aspects, BackColor and Size.

Alphabetic	
BackColor	LightSteelBlue
Font	Arial, 12 pt
Size	405,340
Text	Fibonacci Sequence

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

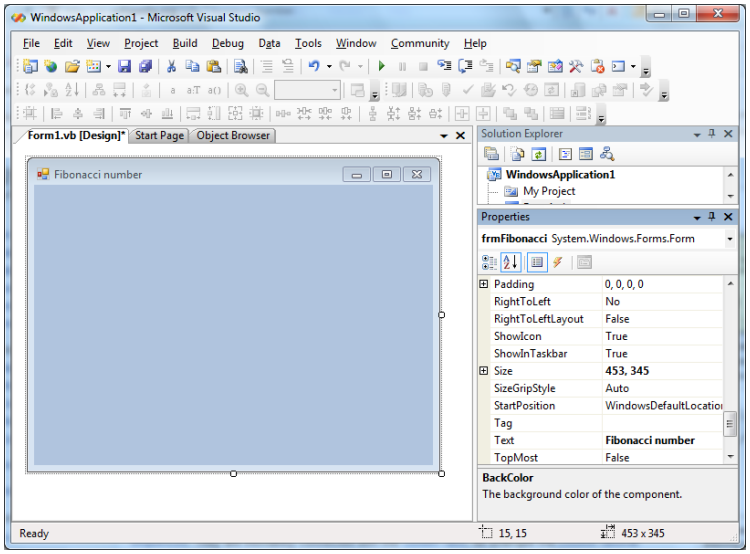

Figure 6.5 – Setting the Form Properties

The background color will change to a Light Steel Blue. 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 Basic Font window.

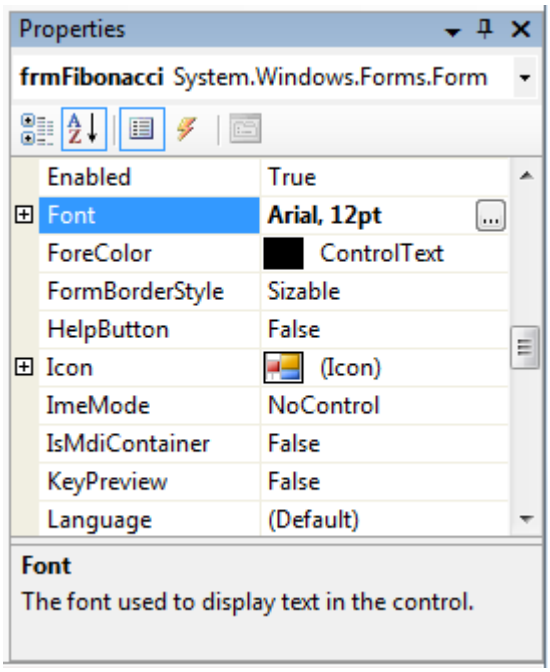

Figure 6.6 – The Font Window in Visual Basic

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.

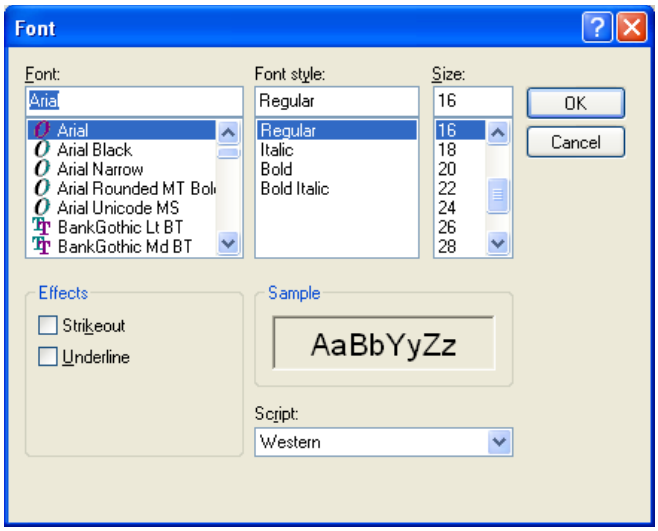

Figure 6.7 – Changing the Font to Arial

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 **lblNumber1.**

Alphabetic	
(Name)	lblNumber1
Text	1st Number

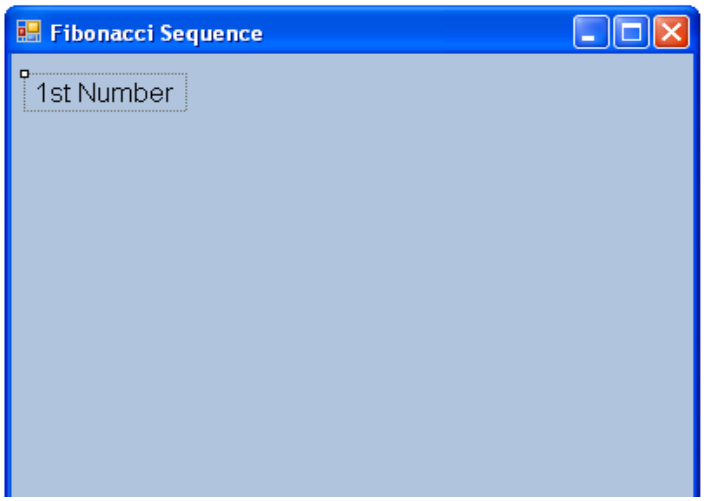

Figure 6.8 – The Finished Label on the Form

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.

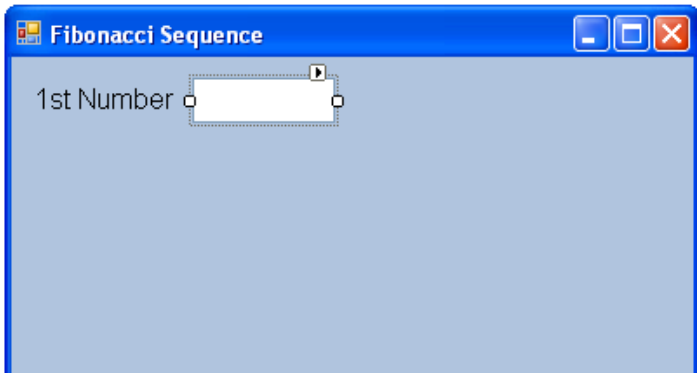

Figure 6.9 – Placing a Textbox on the Form

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 **txtNumber1.**

Alphabetic	
(Name)	txtNumber1
BackColor	White
Size	82,26

The size of the textbox will be 82 wide and 26 tall and the characters inside the textbox will be aligned to the left.

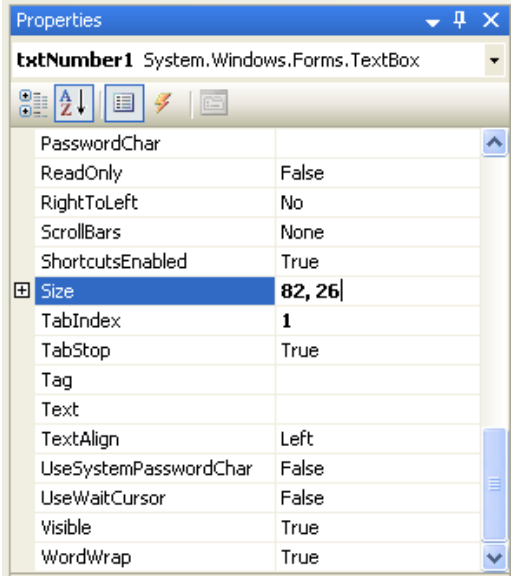

Figure 6.10 – Setting the Size of the Textbox

Inserting another Set of Labels and Textboxes into a Form

We will insert another Label to the right of the textbox and call the entity **lblNumber2.**

Alphabetic	
(Name)	lblNumber1
Text	1st Number

We will insert another textbox to the right of the label and name the new textbox, **txtNumber2.**

Alphabetic	
(Name)	txtNumber1
BackColor	White
Size	82,26

The size of the textbox will be 82 wide and 26 tall and the characters inside the textbox will be aligned in the left.

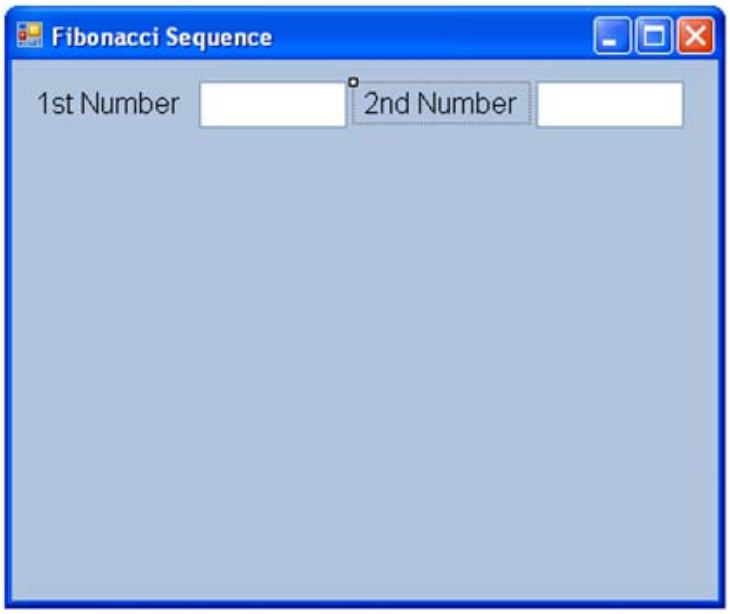

Figure 6.11 – Adding another Label and Textbox

We will insert another Label beneath the first label and call the entity **lblNumbers.**

Alphabetic	
(Name)	lblNumbers
Text	Numbers in the Sequence

We will insert another textbox to the right of the label and name the new textbox, **txtQtyNumbers.**

Alphabetic	
(Name)	txtQtyNumbers
BackColor	White
Size	82,26

The size of the textbox will be 82 wide and 26 tall and the characters inside the textbox will be aligned in the left.

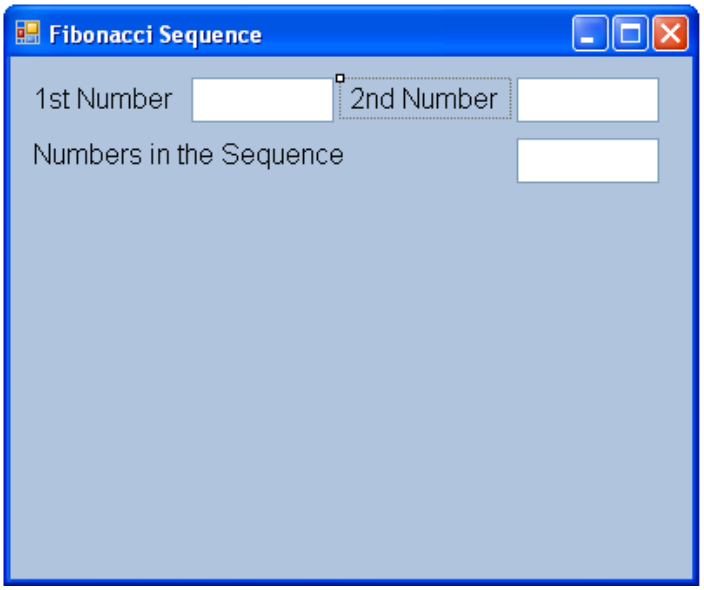

Figure 6.12 – Adding another Label and Textbox

Using a Read Only Textbox with a Vertical Scroll Bar

Instead of using a label to show our sequence of numbers for the answer, we will use a read-only textbox. We insert the textbox beneath the Numbers in the Sequence label and textbox, **txtQtyNumbers**. We call the textbox, **txtSequence**.

Alphabetic	
(Name)	txtSequence
BackColor	White
Mulitline	True
ReadOnly	True
Scrollbar	Vertical
Size	360,140
TextAlign	Right

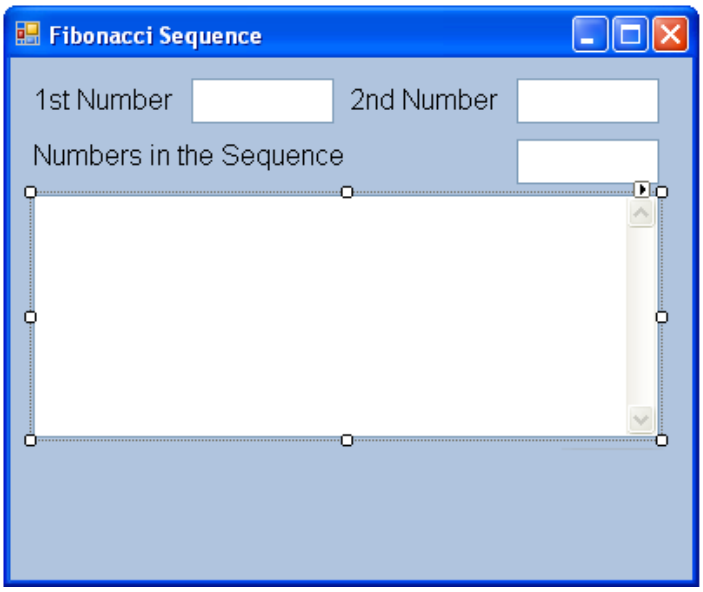

Figure 6.13 – Adding a Read-Only Textbox

We are used to giving the textbox a certain name, size and color. However, in this project, we would like the output to show at least seven numbers in the list. The vertical scroll bar allows the user to pan up and down the list of numbers that we are aligning to the right side of the textbox. We need to be sure that we change the properties and listed above.

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 **cmdCalculate.**

Alphabetic	
(Name)	cmdCalculate
Caption	Calculate
Size	106,33

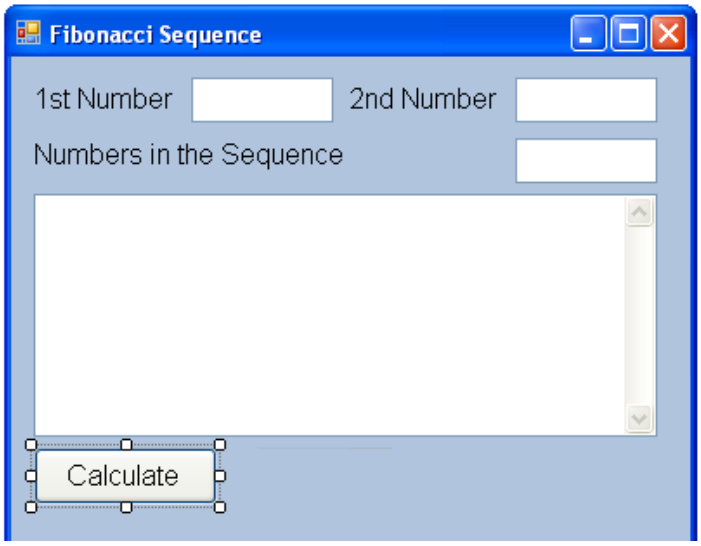

Figure 6.14 – The Command cmdCalculate Button

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.

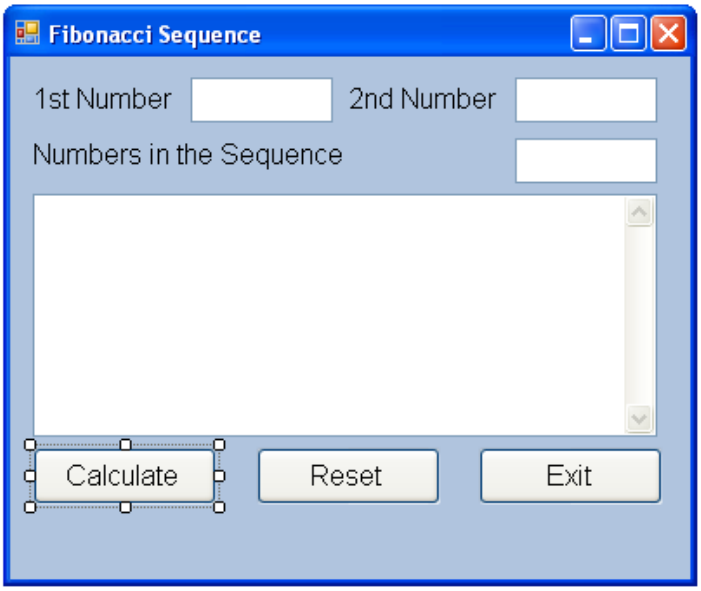

Figure 6.15 – Insert Two More Command Buttons

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.

Fibonacci Generator.dv 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.

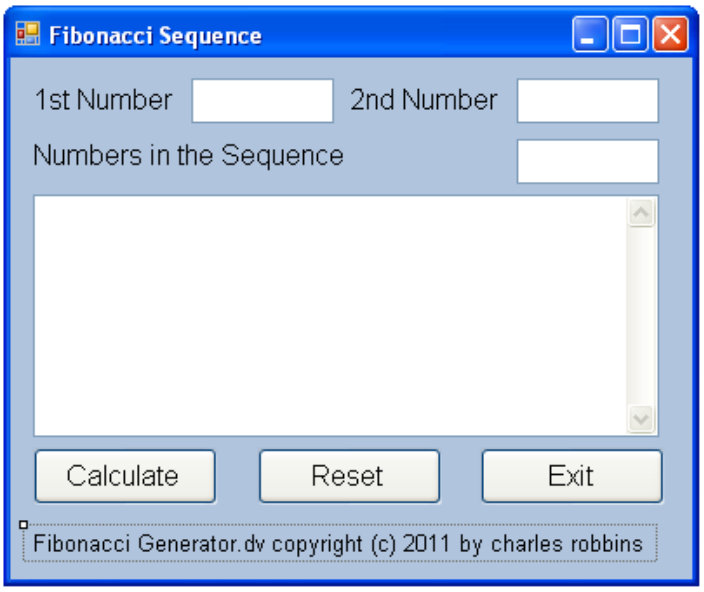

Figure 6.16 – Adding a Copyright Statement

Adding Comments in Visual Basic 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:

```
' Fibonacci Generator.dv copyright (c) 2011 by Charles W. Robbins
' this program will create a fibernacci sequence from two numbers.
```

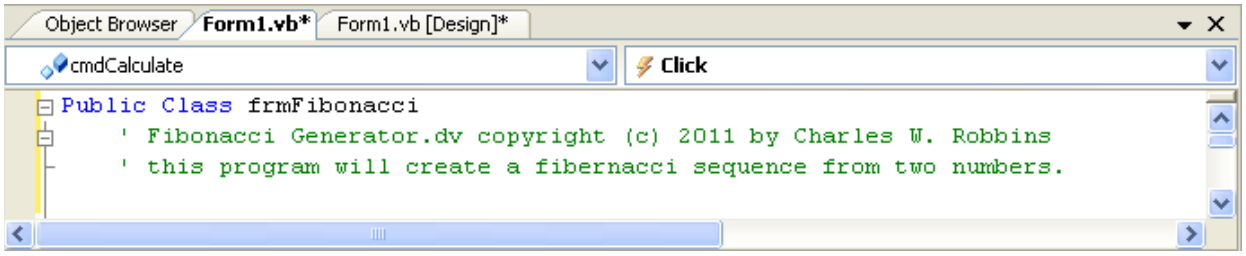

Figure 6.17 – Adding a Copyright Statement

Declaring Variables in a Program with the Dimension Statement

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 Visual Basic, the dimension statement is one of the ways to declare a variable at the procedure level. The other two ways are the Private and Public statements, which we will use in later chapters.

In our program, we will retrieve the data from the textboxes and also we will create data from mathematical computations. We will place the values in variables called Number1, Number2, Number3, Quantity and Counter. These variables will hold numbers for calculations so we will declare them as Double Integers.

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

```
'Declare variable
Dim Number1 As Double
Dim Number2 As Double
Dim Numbers As Double
Dim Quantity As Double
```

```
Dim Counter As Double
Dim Answer As String
```

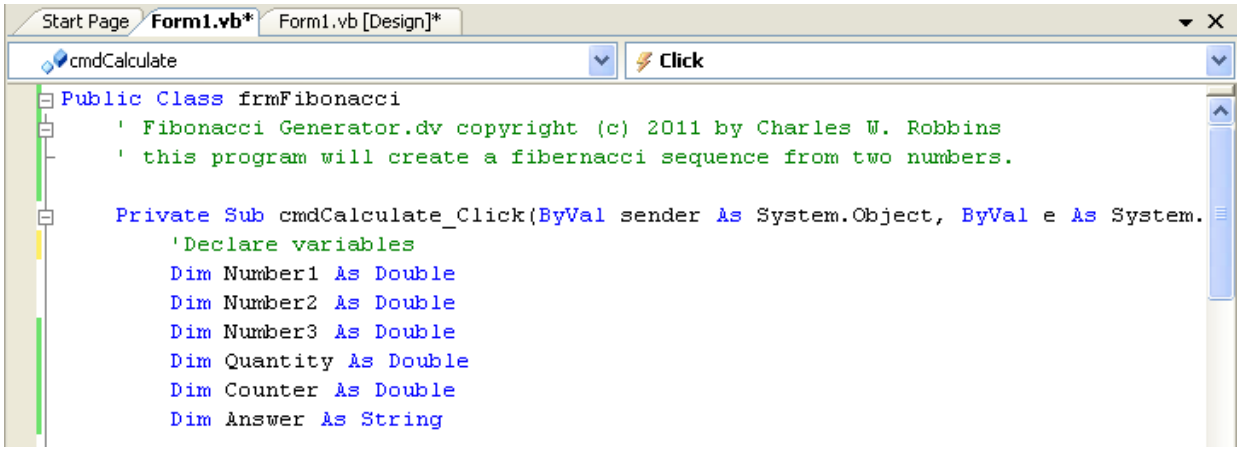

Figure 6.18 – Declaring Variables with Dim Statements

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 numbers in the first number textbox to the variable **Number1** and the second number textbox to the variable **Number2.** We place the value of the numbers in the sequence to the variable **Quantity**. We set the variable counter to zero for the While loop. In our answer, the first two numbers will of course be in there, so before we enter the loop to add additional numbers in the set, we put numbers one and two. We use the Number1.ToString to change the value into a string and we concatenate the phase with the & sign. Each vbCrLf will start a new line for each number.

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

```
'Set variables
Number1 = txtNumber1.Text
Number2 = txtNumber2.Text
Quantity = txtQtyNumbers.Text
Counter = 0
Answer = Number1.ToString & vbCrLf & Number2.ToString
```

```
cmdCalculate                                   Click
Private Sub cmdCalculate_Click(ByVal sender As System.Object, ByVal e As System.
    'Declare variables
    Dim Number1 As Double
    Dim Number2 As Double
    Dim Number3 As Double
    Dim Quantity As Double
    Dim Counter As Double
    Dim Answer As String

    'Set variables
    Number1 = txtNumber1.Text
    Number2 = txtNumber2.Text
    Quantity = txtQtyNumbers.Text
    Counter = 0
    Answer = Number1.ToString & vbCrLf & Number2.ToString
```

Figure 6.19 – Setting the Variables in the Code

Processing Inside a the Loop

To process the loop, we use the Do While function and after the phrase, we type counter is less than quantity minus two.

```
Do While Counter < Quantity - 2
```

Counter is equal to zero and quantity is equal to the amount of numbers in the Fibonacci sequence. We subtract 2 from the variable **Quantity,** because there are already two number in the sequence that we added in the last section of the code (Number1 and Number2).

When we process inside the Do While Loop, we add the two numbers and concatenate the Number3 total to the Answer string using this code.

```
Answer = Answer & vbCrLf & Number3.ToString
```

After that, we change the variable **Number1** to **Number2** and **Number2** to **Number3.** We also add one to the counter. Remember, in Chapter 5, if we do not add one to the counter, the loop will continue forever. Then we close the Do While Loop with Loop. We can see the whole procedure shown below.

```
Do While Counter < Quantity - 2

  'Process
  Number3 = Number1 + Number2
  Answer = Answer & vbCrLf & Number3.ToString
  Number1 = Number2
  Number2 = Number3
  Counter = Counter + 1

Loop
```

```
        Counter = 0
        Answer = Number1.ToString & vbCrLf & Number2.ToString

        Do While Counter < Quantity - 2

            'Process
            Number3 = Number1 + Number2
            Answer = Answer & vbCrLf & Number3.ToString
            Number1 = Number2
            Number2 = Number3
            Counter = Counter + 1

        Loop

        'Output
        txtSequence.Text = Answer

    End Sub
```

Figure 6.20 – Processing the While Loop

Lastly, will write the sequence to the read-only textbox. The variable **Answer** is already a string, so we can write the code as follows.

```
'Output
txtSequence.Text = Answer
```

To finish the project, we need to code for the Reset and exit buttons.

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 four textboxes
txtNumber1.Text = ""
txtNumber2.Text = ""
txtQtyNumbers.Text = ""
txtSequence.Text = ""
```

```
Object Browser | Form1.vb* | Form1.vb [Design]*
frmFibonacci | (Declarations)
    Private Sub cmdReset_Click(ByVal sender As System.Object, ByVal e As System.
        'Reset the four textboxes
        txtNumber1.Text = ""
        txtNumber2.Text = ""
        txtQtyNumbers.Text = ""
        txtSequence.Text = ""
    End Sub
```

Figure 6.21 – Computing the Reset Button by Clearing the Textboxes

Exiting the Program

```
    Private Sub cmdExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) H
        'Unload and exit the program
        Me.Close()
    End Sub
```

Figure 6.22 – 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
   Me.Close()
```

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

```
Public Class frmFibonacci
   ' Fibonacci Generator.dv copyright (c) 2011 by Charles W. Robbins
   ' this program will create a fibernacci sequence from two numbers.

Private Sub cmdCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles cmdCalculate.Click
      'Declare variable
      Dim Number1 As Double
      Dim Number2 As Double
      Dim Number3 As Double
      Dim Quantity As Double
      Dim Counter As Double
      Dim Answer As String

      'Set variables
      Number1 = txtNumber1.Text
```

```
        Number2 = txtNumber2.Text
        Quantity = txtQtyNumbers.Text
        Counter = 0
        Answer = Number1.ToString & vbCrLf & Number2.ToString

        Do While Counter < Quantity - 2

            'Process
            Number3 = Number1 + Number2
            Answer = Answer & vbCrLf & Number3.ToString
            Number1 = Number2
            Number2 = Number3
            Counter = Counter + 1

        Loop

        'Output
        txtSequence.Text = Answer

    End Sub

    Private Sub cmdReset_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles cmdReset.Click
        'Reset the four textboxes
        txtNumber1.Text = ""
        txtNumber2.Text = ""
        txtQtyNumbers.Text = ""
        txtSequence.Text = ""
    End Sub

    Private Sub cmdExit_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
    Handles cmdExit.Click
        Me.Close()
    End Sub

    End Class
```

Running the Program

After noting that the program is saved, press the F5 to run the Fibonacci Generator application. The Fibonacci Generator window will appear on the graphical display as shown in Figure 6.23. Notice the professional appearance and presentation of information in a clean dialogue box.

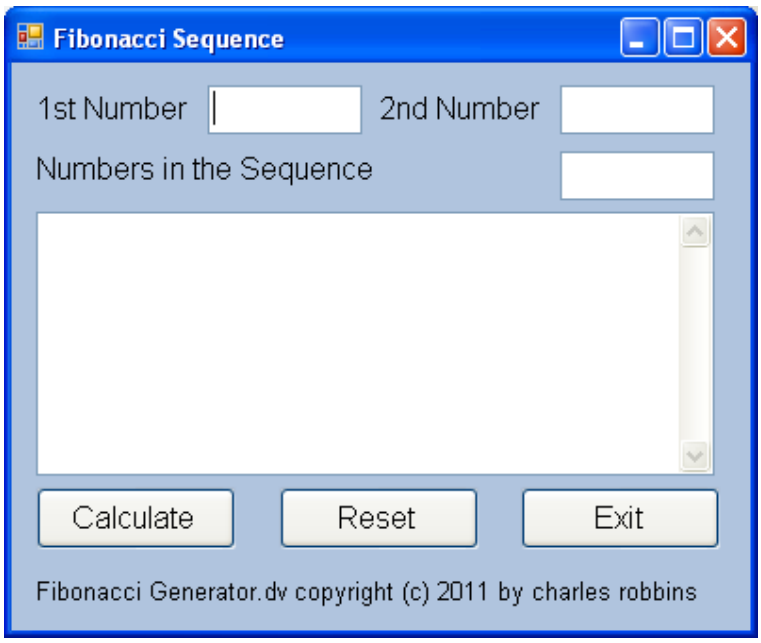

Figure 6.23 – Launching the Program

Type "0" for the first number, "1" for the second number and "5" for the amount of numbers in the sequence as shown in Figure 6.24. If we make a mistake, we can type over the text entry or press the Reset command button to clear the textbox. Press the Calculate command button and five numbers in the Fibonacci sequence will be displayed.

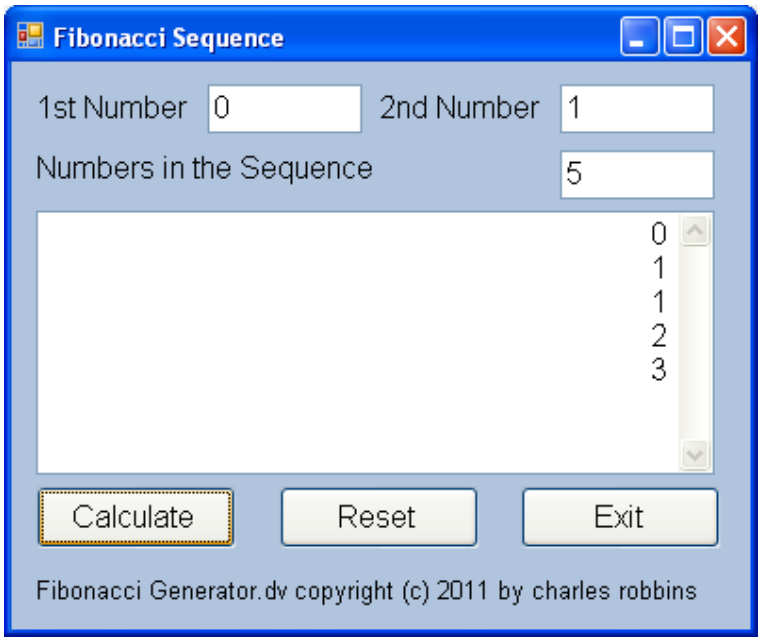

Figure 6.24 – Running the Program

Press the Reset command button to clear the textbox. Type “2” for the first number, “7” for the second number and “10” for the amount of numbers in the sequence as shown in Figure 6.25. Press the Calculate command button and ten numbers in the Fibonacci sequence will be displayed. We will have to use the scroll bar to see the entire list.

To close the program, we press the Exit command button.

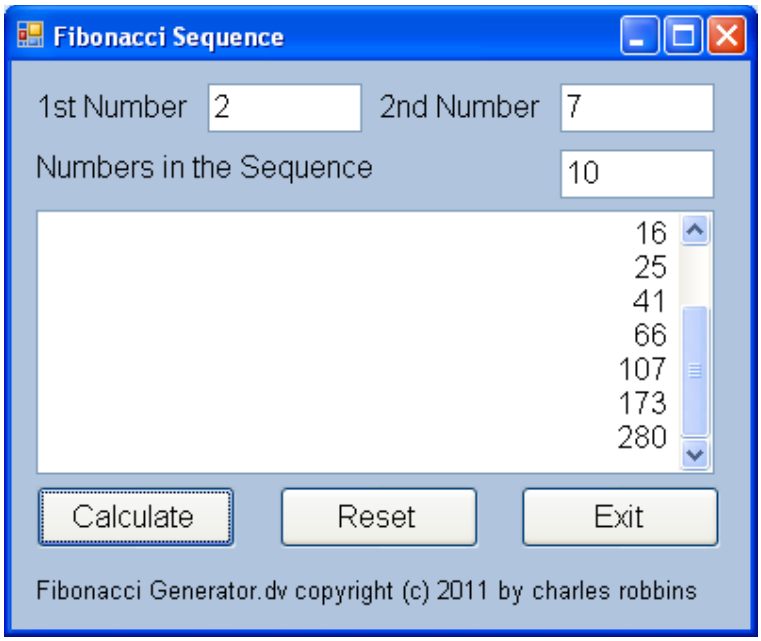

Figure 6.25 – The Sequence is Calculated

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 Basic 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 90-5 * - Write a Visual Basic 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 Basic Project in less than 1 hour to maintain your World Class ranking.