Skills and Topics for KidCoder: Windows Programming

Our Self-Study Approach

Our courses are **self-study** and can be completed on the student's own computer, at their own pace. You can steer your student in the right direction with no prior programming knowledge. Students only need typical computer usage skills to start; we will teach them programming from the ground up!

Each course comes with student activity starters, supplementary instructional documents, a **Solution Guide**, fully coded solutions for all activities, tests and answer keys, and guidance on evaluating projects.

Most questions about how to code individual activities are easily answered by referring to the Solution Guide (with or without parental involvement). We also provide **free technical support** to assist with any aspect of the courses!

Teachers who wish to closely monitor and grade student progress for credit purposes can administer chapter tests which are provided (with answer keys). We also provide advice and guidelines for evaluating student activities.

What Skills do Students Need to Begin?

All of our courses assume the student is already familiar with using a keyboard and mouse to select and run software, navigate the menus in a typical software program, and generally interact with their computer.

Students should understand how to use the built-in operating system software (Windows Explorer or Mac Finder) to find, save and retrieve files on their computer. It may also be helpful to have some familiarity with text editors (like Notepad or TextEdit) and some experience using web browsers to find information on the Internet.

We teach students how to program a computer from the ground up, but they should already know the basics about using one!

Topics Covered In This Course

The following are some of the computer programming topics that are covered in this course. For a full list of topics and sections, please see the Table of Contents for this course.

- Introduction to computer hardware, software and programming history
- Using the Microsoft Visual Basic 2010 Express development environment
- Managing numeric and text data
- Making decisions about program flow
- Obtaining and validating user input
- Working with numbers and math operations
- Working with strings (text)
- Learning how to debug (find errors in) your code
- Learning how to write loops to execute sections of code many times
- Working with arrays (sets of data)
- Publishing your programs to other computers
- Putting it all together – write a simple graphical game!

KidCoder™ Series

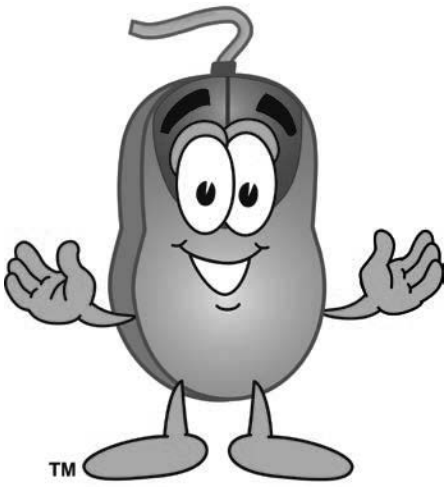
Windows Programming

A hands-on introduction to the field of Windows Programming for elementary or middle-school students.

Student Textbook

Third Edition

KidCoder™ Series



KidCoder™: Windows Programming

Student Textbook

Third Edition

Homeschool Programming, Inc.

KidCoder™: Windows Programming

980 Birmingham Rd, Suite 501-128, Alpharetta, GA 30004

ISBN: **978-0-9830749-9-1**

Terms of Use

Disclaimer

Homeschool Programming, Inc., and their officers and shareholders, assume no liability for damage to personal computers or loss of data residing on personal computers arising due to the use or misuse of this course material. Always follow instructions provided by the manufacturer of 3rd party programs that may be included or referenced by this course.

Contact Us

You may contact Homeschool Programming, Inc. through the information and links provided on our website: http://www.HomeschoolProgramming.com. We welcome your comments and questions regarding this course or other related programming courses you would like to study!

Other Courses

Homeschool Programming, Inc. currently has two product lines for students: the *KidCoder™* series and the *TeenCoder™* series. Our *KidCoder™* series provides easy, step-by-step programming curriculum for 4th through 12th graders. These courses use readily available software products that come shipped with the operating system or are free to install in order to teach introductory programming concepts in a fun, graphical manner. Our *TeenCoder™* series provides introductory programming curriculum for high-school students. These courses are college-preparatory material designed for the student who may wish to pursue a career in Computer Science or enhance their transcript with a technical elective.

3rd Party Copyrights

This course uses Microsoft's Visual Basic 2010 Express as the programming platform. Visual Studio, Visual Studio Express, Windows, and all related products are copyright of Microsoft Corporation. Please see http://www.microsoft.com/visualstudio/eng/products/visual-studio-express-products for more details.

Instructional Videos

This course may be accompanied by optional Instructional Videos. These Flash-based videos will play directly from a DVD drive on the student's computer. Instructional Videos are supplements to the Student Textbook, covering every chapter and lesson with fun, animated re-enforcement of the main topics.

Instructional Videos are intended for students who enjoy a more audio-visual style of learning. They are not replacements for the Student Textbook which is still required to complete this course. However by watching the Instructional Videos first, students may begin each textbook chapter and lesson already having some grasp of the material to be read. Where applicable, the videos will also show "screencasts" of a real programmer demonstrating some concept or activity within the software development environment.

This Student Textbook and accompanying material are entirely sufficient to complete the course successfully. Instructional Videos are optional for students who would benefit from the alternate presentation of the material. For more information or to purchase the videos separately, please refer to the product descriptions on our website: http://www.HomeschoolProgramming.com.

Table of Contents

Before You Begin

Please read the following topics before you begin the course.

Minimum Hardware and Software Requirements

This is a hands-on programming course. You will be installing Microsoft's Visual Basic 2010 Express software on your computer. Your computer must meet the following minimum requirements in order to run Visual Basic 2010 Express:

Computer Hardware

Your computer must meet the following minimum specifications:

	Minimum
CPU	1.6GHz or faster processor
RAM	1024 MB
Display	1024 x 768 compatible video card
Hard Disk Size	3GB available space
DVD Drive	DVD-ROM drive

Operating Systems

In order to install the course software, your computer operating system must match one of the following:

Windows XP (x86) with Service Pack 3 or above
Windows Vista (x86 and x64) with Service Pack 2 or above
Windows 7 (x86 and x64)
Windows 8 (all versions except RT)

Conventions Used in This Text

This course will use certain styles (fonts, borders, etc.) to highlight text of special interest.

```
Source code will be in 11-point Consolas font, in a single box like this.
```

Variable names will be in **`12-point Consolas bold`** text. For example: **`myVariable`**.

Function and subroutine names, properties, and keywords will be in **bold face** type.

This picture highlights important concepts within a lesson.

Sidebars may contain additional information, tips, or background material.

A chapter review section is included at the end of each chapter.

Every chapter includes a "Your Turn" activity that allows you to practice the ideas you have learned.

What You Will Learn and Do In This Course

KidCoder™: Windows Programming will teach you the basics of writing your own computer programs. Along the way, you will begin to understand the building blocks for other applications that you may use every day.

Starting with Chapter Two, each chapter will include a walk-through of a program that demonstrates the concepts that you are learning. You will build this program in your development environment on your own computer to see it work in real life. At the end of each chapter, you will get a chance to add something to each program on your own.

What You Need to Know Before Starting

You are expected to already know the basics of computer use before beginning this course. You need to know how to use the keyboard and mouse to select and run programs, use application menu systems, and work with the Windows operating system. You should understand how to store and load files on your computer and how to use Windows Explorer to walk through your file system and directory structures. You should also have some experience with using text editors and using web browsers to find helpful information on the Internet.

Software Versions

You will be using the *Microsoft Visual Basic 2010 Express* software to complete this course. This program can be freely downloaded from Microsoft's website. Our website contains download and install instructions for this software, and your course Student Menu contains a direct link to this page. All supplemental documents are in Adobe Acrobat (PDF) format. You must have the Adobe Acrobat Reader installed to view these documents.

Course Errata

We welcome your feedback regarding any course details that are unclear or that may need correction. You can find a list of course errata for this edition on our website.

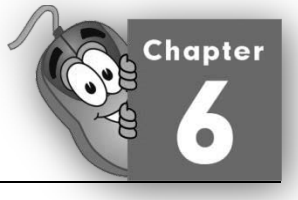
Getting Help

All courses come with a Solution Guide and fully coded solutions for all activities. Simply install the "Solution Files" from your course setup program and you will be able to refer to the solutions as needed from the "Solution Menu". If you are confused about any activity, this will allow you to see how we solved the problem!

We also offer free technical support for students and teachers. Simply fill out the help request form in the "Support" area of our website with a detailed question and we will assist you.

SAMPLE STUDENT LESSON

The following pages contain a sample student lesson from the KidCoder: Windows Programming textbook.



Chapter Six: Getting User Input

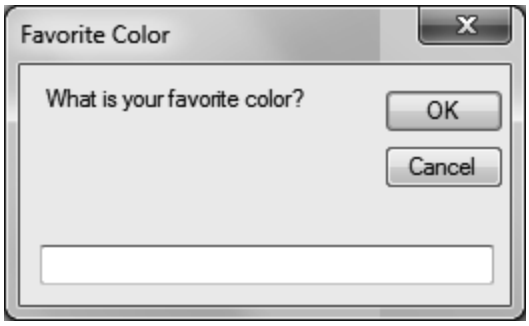
User input is an important part of any program. This chapter will explain how to ask for, receive, and validate user input.

Lesson One: InputBoxes

So far we have used the Visual Basic **MsgBox**() pop-up window to display output to the user. In this lesson, we will learn how to use the **InputBox**() pop-up window to receive input.

The **InputBox**() in Visual Basic is a special function which can be used to get a single line of input text from a user. For example, if you needed to know a user's age, or you needed to know their home state, you could use an **InputBox**().

When you use the **InputBox**() function, the program will show a simple window with a question for the user. The window will also have a place for the user to type in their answer, an "OK" button and a "Cancel" button. If the user clicks the "OK" button, any information typed into the text box will be sent back to your program. If the user clicks "Cancel", they will return to your program without sending any information.



Here is what the **InputBox**() syntax looks like:

```
stringVariable = InputBox("question for user", "title for window")
```

Let's look at this statement one part at a time. The first **stringVariable** is the name of some **String** variable that you must have already declared. When the user clicks the "OK" button, Visual Basic will put the use text from the edit field into this variable. If the user clicks the "Cancel" button, this variable will contain an empty string.

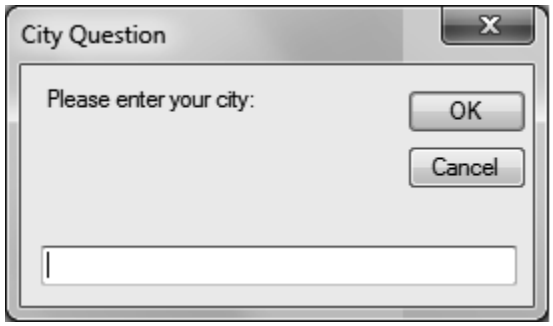
The next part is the **InputBox**() function name. This just tells Visual Basic that we will be showing an input box pop-up window. The items within the parentheses are the parameters for our **InputBox**(). The first parameter, "question for user", is just a string that tells the user what information we will need. For example, if we need to ask the user to enter their city, we could use the string: "Please enter your city:"

The second parameter, "title for window", allows us to give our **InputBox**() a useful caption or title at the top. Continuing the example above, we might use "City Question" as our title.

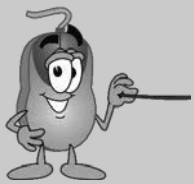
Now let's look at a complete example that asks the user to enter the name of his or her city:

```
Private Sub Button1_Click(...

  Dim city As String
  city = InputBox("Please enter your city: ", "City Question")

End Sub
```

Here we have assumed that you have a Form with a Button on it, and when the user click on the Button we want to run our code to ask the user to enter the city name. When run, the **InputBox**() pop-up window would look like this:



The value the user types into the edit field at the bottom will be stored in the city variable when the user clicks on the "OK" button. That was pretty easy, right?



There are many more parameters available for the InputBox() function. For this course, we will only be using the two parameters mentioned above. If you are interested in learning about the other parameters, just look up the InputBox() function in your MSDN Help library.

Lesson Two: Getting User Input from Forms

The **InputBox**() function is a quick and simple way to get a single piece of information from a user. But what if you need more than one piece of information? What if you need to get a person's whole address and not just their city name? For this, you could create a custom Form that has several textboxes for users to enter information. Your form could also have an "OK" button the user can click when finished.

Let's walk through the creation of a custom input form together. Start your Microsoft Visual Basic IDE and create a new project called "User Input". Once you have created your project, we need to rename and title the Form. Click on the Form and then look at the Properties window. Find the property called **(Name)** and change it from "Form1" to "MyInputForm". Then find the property called **Text** and change it from "Form1" to "Address Form".

The TextBox Control

Now, look at your blank Form and the Toolbox. For our address form, we will need four pieces of information from the user: a street address, city, state and zip code. We will be using four TextBox controls to hold this information and four Label controls to describe the TextBoxes. Go ahead and drag and drop four Labels and four TextBoxes onto your Form.

Carefully line up one Label with each TextBox. Next we need to give our labels more meaningful descriptions. Click on Label1 and then look at the Properties window. Find the property called **Text** and change the text from "Label1" to "Street:" Then click on Label2 and do the same thing to change the text from "Label2" to "City:" Next, change the text for Label3 to "State:", and the text for Label4 to "Zip code:". Your new form is shown on the left.

We will also need to change the name of our TextBoxes. To do this, click on the first TextBox and find the property called **(Name)** in the Properties window. Change the name from "Textbox1" to "StreetTextbox". Then click on the second TextBox and do the same thing to change the **(Name)** from "Textbox2" to "CityTextbox". Next, change the **(Name)** for Textbox3 to "StateTextbox" and the **(Name)** for "Textbox4" to "ZipTextbox".

In just a minute, you will use the **(Name)** of each control to read that control from your Visual Basic code!

Now we need to add an "OK" Button to our Form. Drag and drop a Button onto your Form and put it at the bottom of the screen. Then rename the Button control and change the display text. Click on "Button1" and then look at the Properties window. Change the **(Name)** property from "Button1" to "OKButton" and the "Text" property from "Button1" to "OK". Your final Form is shown on the left.

Great! We have a Form that will ask a user for full address information. Now let's talk about how we are going to get that information from the controls when the user clicks the "OK" button.

Reading TextBox Data

You can read TextBox contents from the **Text** property attached to the Textbox name. In the example below, we read the value from the **StreetTextbox** control and store it in a new **String** called **street**:

```
Dim street As String = StreetTextbox.Text
```

Once the user clicks the "OK" button, we can look at each Textbox **Text** property to see what they have entered. For this example, we will use this data in a pop-up window echoing the text back to the user.

Double-click on the "OK" Button to create a click event subroutine. You should now see the code window with the click event function. We are going to add our new code inside the **OKButton_Click**() subroutine.

```
Public Class MyInputForm
    Private Sub OKButton_Click(ByVal sender As System.Object, _
                               ByVal e As System.EventArgs) Handles OKButton.Click
    End Sub
End Class
```

First, declare new **String** variables to hold the address data. Place your cursor within the function (after the "Private Sub..." line) and create four **String** variables:

```
Dim streetString As String
Dim cityString As String
Dim stateString As String
Dim zipString As String
```

Now we have created the strings which will hold our user's data. But how do we get the data into these strings? In order to get the information from the textboxes, we will read the **Text** property of each Textbox. Each Textbox can be referred to by the **(Name)** property we gave it on the Form. To read a value from

the Textbox, use the name of the Textbox, a period, and then the **Text** property name. Add these four lines to get the text data from the four textboxes into our string variables:

```
streetString = StreetTextbox.Text
cityString = CityTextbox.Text
stateString = StateTextbox.Text
zipString = ZipTextbox.Text
```

Notice how we set our **String** variables equal to the name of each TextBox with a "**.Text**" afterwards. This tells our program to take the text that is currently in that TextBox and place it into our **String** variable.

Your code so far should look something like this:

```
Private Sub OKButton_Click(ByVal sender As System.Object, _
                           ByVal e As System.EventArgs) _
                           Handles OKButton.Click
    Dim streetString As String
    Dim cityString As String
    Dim stateString As String
    Dim zipString As String

    streetString = StreetTextbox.Text
    cityString = CityTextbox.Text
    stateString = StateTextbox.Text
    zipString = ZipTextbox.Text
End Sub
```

Now that will work to read all of the user's information, but what do we do with it? Let's echo the information back out using a **MsgBox**() command to show the user what they entered. Enter one more line with the **MsgBox**() function, combining all of the input strings into a single message:

```
MsgBox("You entered: " & streetString & " " & cityString & " " & _
           stateString & " " & zipString)
```

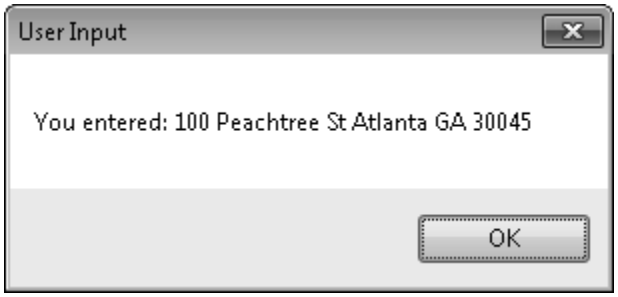
Notice we used the ampersand (&) character to combine our strings together into one big string. Also, note that we used an underscore (_) at the end of the first line to show that we broke the statement across two lines. If you fit the whole statement on one line in your IDE, you do not need this character. Also notice that we added a space " " between each string in order to separate the variable data in the displayed message.

Now your final "OK" Button code should look like this:

```
Private Sub OKButton_Click(ByVal sender As System.Object, _
                           ByVal e As System.EventArgs) _
                           Handles OKButton.Click
    Dim streetString As String
    Dim cityString As String
    Dim stateString As String
    Dim zipString As String

    streetString = StreetTextbox.Text
    cityString = CityTextbox.Text
    stateString = StateTextbox.Text
    zipString = ZipTextbox.Text

    MsgBox("You entered: "& streetString & " " & cityString & " " & _
          stateString & " " & zipString)
End Sub
```

Now run the program and see how it works! Type in some data for all 4 fields and click the "OK" button.

Your popup should display the combined text.



Remember to save your project when you close the application. Name your project "User Input", make sure you are saving in your "C:\KidCoder\Windows Programming\My Projects" directory, and name your solution "User Input".

SAMPLE SOLUTION GUIDE

The following pages contain sample solution material for an activity in the KidCoder: Windows Programming textbook.

Chapter Three Activity (A Personal Message)

In this activity the student will open the "Hello World2" project they created during the chapter and make the following changes:

- Add another **MsgBox** after the one in the program.
- Change the text in your new **MsgBox** to say your name

To add the second **MsgBox**, double-click on the "Click Here" button. Just below the line "**MsgBox("Hello, World!**")", create a new line that looks something like this:

```
MsgBox("My name is Joe Smith")
```

Note: Make sure that your student uses their own name!

The final code should look something like this:

```
Public Class MyForm

    Private Sub MyButton_Click(ByVal sender As System.Object, ByVal e As _
                               System.EventArgs) Handles MyButton.Click
        MsgBox("Hello, World!")

        MsgBox("My name is Joe Smith")

    End Sub
End Class
```

The completed project for this activity is located in the "Your Turn Solutions\Hello World2" folder underneath the Solution Files installation directory.