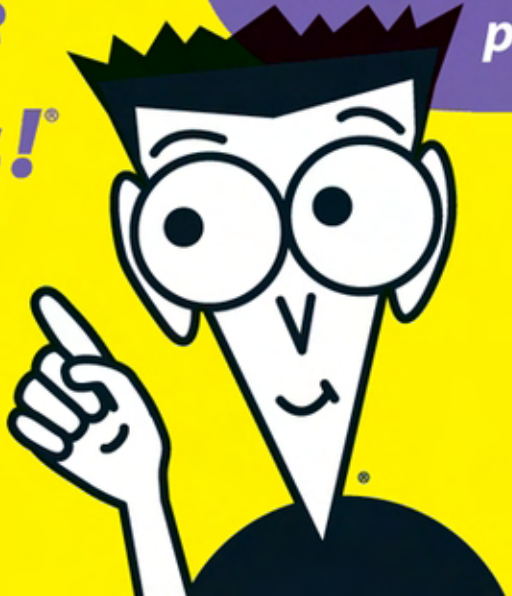
Unleash the power of Excel 2007 with Visual Basic for Applications

Excel 2007 VBA Programming FOR DUMMIES®

Companion Web site includes useful sample programs

A Reference for the Rest of Us!®

FREE eTips at dummies.com®



John Walkenbach
Author of Excel 2003 Bible

Excel® 2007 VBA Programming FOR DUMMIES®

Chapter 1: What Is VBA?

ISBN-10: 0-470-04674-0
ISBN-13: 978-0-470-04674-6

by John Walkenbach
Revised by Jan Karel Pieterse

Wiley Publishing, Inc.

Excel® 2007 VBA Programming For Dummies®
Published by
Wiley Publishing, Inc.
111 River Street
Hoboken, NJ 07030-5774
www.wiley.com

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

Library of Congress Control Number: 2006939593

ISBN: 978-0-470-04674-6

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

About the Author

John Walkenbach is the author of more than 50 spreadsheet books and lives in southern Arizona. Visit his Web site at `http://j-walk.com`.

Dedication

"This book is dedicated to Jim Kloss and Esther Golton — my two favorite people in Matanuska-Susitna county. By putting their names in this book, I'm ensured of at least one sale in Alaska."

Author's Acknowledgments

Thanks to all of the talented people at Wiley Publishing for making it so easy to write these books. And special thanks to Jan Karel Pieterse for his assistance with this edition.

Publisher's Acknowledgments

We're proud of this book; please send us your comments through our online registration form located at www.dummies.com/register/.

Some of the people who helped bring this book to market include the following:

Acquisitions, Editorial, and Media Development

Project Editor: Beth Taylor

Executive Editor: Greg Croy

Copy Editor: Beth Taylor

Technical Editor: Allen Wyatt

Editorial Manager: Jodi Jensen

Media Development Coordinator: Laura Atkinson

Media Project Supervisor: Laura Moss

Media Development Manager: Laura VanWinkle

Media Development Associate Producer: Richard Graves

Editorial Assistant: Amanda Foxworth

Sr. Editorial Assistant: Cherie Case

Cartoons: Rich Tennant (www.the5thwave.com)

Composition Services

Project Coordinator: Jennifer Theriot

Layout and Graphics: Carl Byers, Stephanie D. Jumper, Barbara Moore, Julie Trippetti

Proofreaders: Laura Albert, John Greenough, Techbooks

Indexer: Techbooks

Publishing and Editorial for Technology Dummies

Richard Swadley, Vice President and Executive Group Publisher

Andy Cummings, Vice President and Publisher

Mary Bednarek, Executive Acquisitions Director

Mary C. Corder, Editorial Director

Publishing for Consumer Dummies

Diane Graves Steele, Vice President and Publisher

Joyce Pepple, Acquisitions Director

Composition Services

Gerry Fahey, Vice President of Production Services

Debbie Stailey, Director of Composition Services

Introduction

Greetings, prospective Excel programmer . . .

Thanks for buying my book. I think you'll find that it offers a fast, enjoyable way to discover the ins and outs of Microsoft Excel programming. Even if you don't have the foggiest idea of what programming is all about, this book can help you make Excel jump through hoops in no time (well, it will take *some* time).

Unlike most programming books, this one is written in plain English, and even normal people can understand it. Even better, it's filled with information of the "just the facts, ma'am" variety — and not the drivel you might need once every third lifetime.

Is This the Right Book?

Go to any large bookstore and you'll find many Excel books (far too many, as far as I'm concerned). A quick overview can help you decide whether this book is really right for you. This book

- Is designed for intermediate to advanced Excel users who want to master Visual Basic for Applications (VBA) programming.
- Requires no previous programming experience.
- Covers the most commonly used commands.
- Is appropriate for Excel 2007.
- Just might make you crack a smile occasionally — it even has cartoons.

If you are using Excel 2000, XP, or 2003, this book is not for you. Excel 2007 is so different from previous versions. If you're still using a pre-2007 version of Excel, locate a book that is specific to that version.

This is *not* an introductory Excel book. If you're looking for a general-purpose Excel book, check out any of the following books, which are all published by Wiley:

- *Excel 2007 For Dummies,* by Greg Harvey
- *Excel 2007 Bible,* by John Walkenbach (yep, that's me)
- *Excel 2007 For Dummies Quick Reference,* by John Walkenbach (me again) and Colin Banfield

Notice that the title of this book isn't *The Complete Guide to Excel VBA Programming For Dummies*. I don't cover all aspects of Excel programming — but then again, you probably don't want to know *everything* about this topic. In the unlikely event that you want a more comprehensive Excel programming book, you might try *Microsoft Excel 2007 Power Programming With VBA,* by John Walkenbach (is this guy prolific, or what?), also published by Wiley.

So You Want to Be a Programmer . . .

Besides earning money to pay my bills, my main goal in writing this book is to show Excel users how to use the VBA language — a tool that helps you significantly enhance the power of the world's most popular spreadsheet. Using VBA, however, involves programming. (Yikes! The *p* word.)

If you're like most computer users, the word *programmer* conjures up an image of someone who looks and behaves nothing like you. Perhaps words such as *nerd, geek,* and *dweeb* come to mind.

Times have changed. Computer programming has become much easier, and even so-called normal people now engage in this activity. *Programming* simply means developing instructions that the computer automatically carries out. *Excel programming* refers to the fact that you can instruct Excel to automatically do things that you normally do manually — saving you lots of time and (you hope) reducing errors. I could go on, but I need to save some good stuff for Chapter 1.

If you've read this far, it's a safe bet that you need to become an Excel programmer. This could be something you came up with yourself or (more likely) something your boss decided. In this book, I tell you enough about Excel programming so that you won't feel like an idiot the next time you're trapped in a conference room with a group of Excel aficionados. And by the time you finish this book, you can honestly say, "Yeah, I do some Excel programming."

Why Bother?

Most Excel users never bother to explore VBA programming. Your interest in this topic definitely places you among an elite group. Welcome to the fold! If you're still not convinced that mastering Excel programming is a good idea, I've come up with a few good reasons why you might want to take the time to learn VBA programming.

- **It will make you more marketable.** Like it or not, Microsoft's applications are extremely popular. You may already know that all applications in Microsoft Office support VBA. The more you know about VBA, the better your chances for advancement in your job.
- **It lets you get the most out of your software investment (or, more likely, your *employer's* software investment).** Using Excel without knowing VBA is sort of like buying a TV set and watching only the odd-numbered channels.
- **It will improve your productivity (eventually).** Mastering VBA definitely takes some time, but you'll more than make up for this in the amount of time you ultimately save because you're more productive. Sort of like what they told you about going to college.
- **It's fun (well, sometimes).** Some people really enjoy making Excel do things that are otherwise impossible. By the time you finish this book, you just might be one of those people.

Now are you convinced?

What I Assume about You

People who write books usually have a target reader in mind. For this book, my target reader is a conglomerate of dozens of Excel users I've met over the years (either in person or out in cyberspace). The following points more or less describe my hypothetical target reader:

- You have access to a PC at work — and probably at home.
- You're running Excel 2007.
- You've been using computers for several years.
- You use Excel frequently in your work, and you consider yourself to be more knowledgeable about Excel than the average bear.
- You need to make Excel do some things that you currently can't make it do.
- You have little or no programming experience.
- You understand that the Help system in Excel can actually be useful. Face it, this book doesn't cover everything. If you get on good speaking terms with the Help system, you'll be able to fill in some of the missing pieces.
- You need to accomplish some work, and you have a low tolerance for thick, boring computer books.

Obligatory Typographical Conventions Section

All computer books have a section like this. (I think some federal law requires it.) Read it or skip it.

Sometimes, I refer to key combinations — which means you hold down one key while you press another. For example, Ctrl+Z means you hold down the Ctrl key while you press Z.

For menu commands, I use a distinctive character to separate menu items. For example, you use the following command to open a workbook file:

File➪Open

Note, that in Excel 2007, there is no such thing as a "File" menu visible on your screen. In fact the File menu has been replaced with the Office button, a little round contraption that shows up on the top-left side of any Office application that has implemented what is called the Ribbon. Any text you need to enter appears in **bold**. For example, I might say, enter **=SUM(B:B)** in cell A1.

Excel programming involves developing *code* — that is, the instructions Excel follows. All code in this book appears in a monospace font, like this:

```
Range("A1:A12").Select
```

Some long lines of code don't fit between the margins in this book. In such cases, I use the standard VBA line continuation character sequence: a space followed by an underscore character. Here's an example:

```
Selection.PasteSpecial Paste:=xlValues, _
   Operation:=xlNone, SkipBlanks:=False, _
   Transpose:=False
```

When you enter this code, you can type it as written or place it on a single line (omitting the spaces and the underscore characters).

Check Your Security Settings

It's a cruel world out there. It seems that some scam artist is always trying to take advantage of you or cause some type of problem. The world of computing is equally cruel. You probably know about computer viruses, which can

cause some nasty things to happen to your system. But did you know that computer viruses can also reside in an Excel file? It's true. In fact, it's relatively easy to write a computer virus by using VBA. An unknowing user can open an Excel file and spread the virus to other Excel workbooks.

Over the years, Microsoft has become increasingly concerned about security issues. This is a good thing, but it also means that Excel users need to understand how things work. You can check Excel's security settings by using the File➪Excel Options➪Trust Center➪Trust Center Settings command. There is a plethora of options in there. If you click the Macro Settings tab, your options are:

- **Disable all macros without notification:** Macros will not work, regardless of what you do.
- **Disable all macros with notification:** When you open a workbook with macros you will either see the Message Bar open with an option you can click to enable macros, or (if the VBE is open), you'll get a message asking if you want to enable macros.
- **Disable all macros except digitally signed macros:** Only macros with a digital signature are allowed to run (but even for those signatures you haven't marked as trusted you still get the security warning).
- **Enable all macros (not recommended; potentially dangerous code can run).**

Consider this scenario: You spend a week writing a killer VBA program that will revolutionize your company. You test it thoroughly, and then send it to your boss. He calls you into his office and claims that your macro doesn't do anything at all. What's going on? Chances are, your boss's security setting does not allow macros to run. Or, maybe he chose to disable the macros when he opened the file.

Bottom line? Just because an Excel workbook contains a macro, it is no guarantee that the macro will ever be executed. It all depends on the security setting and whether the user chooses to enable or disable macros for that file.

In order to work with this book, you will need to enable macros for the files you work with. My advice is to use the second security level. Then when you open a file that you've created, you can simply enable the macros. If you open a file from someone you don't know, you should disable the macros and check the VBA code to ensure that it doesn't contain anything destructive or malicious.

How This Book Is Organized

I divided this book into six major parts, each of which contains several chapters. Although I arranged the chapters in a fairly logical sequence, you can read them in any order you choose. Here's a quick preview of what's in store for you.

Part I: Introducing VBA

Part I has but two chapters. I introduce the VBA language in the first chapter. In Chapter 2, I let you get your feet wet right away by taking you on a hands-on guided tour.

Part II: How VBA Works with Excel

In writing this book, I assume that you already know how to use Excel. The four chapters in Part II give you a better grasp on how VBA is implemented in Excel. These chapters are all important, so I don't recommend skipping past them, okay?

Part III: Programming Concepts

The eight chapters in Part III get you into the nitty-gritty of what programming is all about. You may not need to know all this stuff, but you'll be glad it's there if you ever do need it.

Part IV: Communicating with Your Users

One of the coolest parts of programming in Excel is designing custom dialog boxes (well, at least I like it). The chapters in Part IV show you how to create dialog boxes that look like they came straight from the software lab at Microsoft.

Part V: Putting It All Together

The chapters in Part VI pull together information from the preceding chapters. You discover how to include your own custom buttons in the Excel user interface, you find out how to develop custom worksheet functions, create add-ins, design user-oriented applications, and even work with other Office applications.

Part VI: The Part of Tens

Traditionally, books in the *For Dummies* series contain a final part that consists of short chapters with helpful or informative lists. Because I'm a sucker for tradition, this book has two such chapters that you can peruse at your convenience. (If you're like most readers, you'll turn to this part first.)

Marginal Icons

Somewhere along the line, a market research company must have shown that publishers can sell more copies of their computer books if they add icons to the margins of those books. *Icons* are those little pictures that supposedly draw your attention to various features, or help you decide whether something is worth reading.

I don't know if this research is valid, but I'm not taking any chances. So here are the icons you encounter in your travels from front cover to back cover:

When you see this icon, the code being discussed is available on the Web. Download it, and eliminate lots of typing. See "Get the Sample Files" below, for more information.

This icon flags material that you might consider technical. You may find it interesting, but you can safely skip it if you're in a hurry.

Don't skip information marked with this icon. It identifies a shortcut that can save you lots of time (and maybe even allow you to leave the office at a reasonable hour).

This icon tells you when you need to store information in the deep recesses of your brain for later use.

Read anything marked with this icon. Otherwise, you may lose your data, blow up your computer, cause a nuclear meltdown — or maybe even ruin your whole day.

Get the Sample Files

This book has its very own Web site where you can download the example files discussed and view Bonus Chapters. To get these files, point your Web browser to:

`www.dummies.com/go/excel2007vba.`

Having the sample files will save you a lot of typing. Better yet, you can play around with them and experiment with various changes. In fact, I highly recommend playing around with these files. Experimentation is the best way to master VBA.

Now What?

Reading this introduction was your first step. Now, it's time to move on and become a programmer (there's that *p* word again!).

If you're a programming virgin, I strongly suggest that you start with Chapter 1 and progress in chapter order until you've discovered enough. Chapter 2 gives you some immediate hands-on experience, so you have the illusion that you're making quick progress.

But it's a free country (at least it was when I wrote these words); I won't sic the Computer Book Police on you if you opt to thumb through randomly and read whatever strikes your fancy.

I hope you have as much fun reading this book as I did writing it.

Chapter 1

What Is VBA?

In This Chapter

- Gaining a conceptual overview of VBA
- Finding out what you can do with VBA
- Discovering the advantages and disadvantages of using VBA
- Taking a mini-lesson on the history of Excel

This chapter is completely devoid of any hands-on training material. It does, however, contain some essential background information that assists you in becoming an Excel programmer. In other words, this chapter paves the way for everything else that follows and gives you a feel for how Excel programming fits into the overall scheme of the universe.

Okay, So What Is VBA?

VBA, which stands for *Visual Basic for Applications,* is a programming language developed by Microsoft — you know, the company that's run by the richest man in the world. Excel, along with the other members of Microsoft Office 2007, includes the VBA language (at no extra charge). In a nutshell, VBA is the tool that people like you and me use to develop programs that control Excel.

Imagine an intelligent robot that knows all about Excel. This robot can read instructions, and it can also operate Excel very fast and accurately. When you want the robot to do something in Excel, you write up a set of robot instructions by using special codes. Tell the robot to follow your instructions, while you sit back and drink a glass of lemonade. That's kind of what VBA is all about — a code language for robots. Note, however, that Excel does not come with a robot or lemonade.

A few words about terminology

Excel programming terminology can be a bit confusing. For example, VBA is a programming language, but it also serves as a macro language. What do you call something written in VBA and executed in Excel? Is it a *macro* or is it a *program?* Excel's Help system often refers to VBA procedures as *macros,* so I use that terminology. But I also call this stuff a *program.*

I use the term *automate* throughout this book. This term means that a series of steps are completed automatically. For example, if you write a macro that adds color to some cells, prints the worksheet, and then removes the color, you have *automated* those three steps.

By the way, *macro* does not stand for **M**essy **A**nd **C**onfusing **R**epeated **O**peration. Rather, it comes from the Greek *makros,* which means large — which also describes your paycheck after you become an expert macro programmer.

Don't confuse VBA with VB (which stands for Visual Basic). VB is a programming language that lets you create standalone executable programs (those EXE files). Although VBA and VB have a lot in common, they are different animals.

What Can You Do with VBA?

You're probably aware that people use Excel for thousands of different tasks. Here are just a few examples:

- Keeping lists of things such as customer names, students' grades, or holiday gift ideas (a nice fruitcake would be lovely)
- Budgeting and forecasting
- Analyzing scientific data
- Creating invoices and other forms
- Developing charts from data
- Yadda, yadda, yadda

The list could go on and on, but I think you get the idea. My point is simply that Excel is used for a wide variety of things, and everyone reading this book has different needs and expectations regarding Excel. One thing virtually every reader has in common is the *need to automate some aspect of Excel.* That, dear reader, is what VBA is all about.

For example, you might create a VBA program to format and print your month-end sales report. After developing and testing the program, you can execute the macro with a single command, causing Excel to automatically perform many time-consuming procedures. Rather than struggle through a tedious sequence of commands, you can grab a cup of joe and let your computer do the work — which is how it's supposed to be, right?

In the following sections, I briefly describe some common uses for VBA macros. One or two of these may push your button.

Inserting a bunch of text

If you often need to enter your company name, address, and phone number in your worksheets, you can create a macro to do the typing for you. You can extend this concept as far as you like. For example, you might develop a macro that automatically types a list of all salespeople who work for your company.

Automating a task you perform frequently

Assume you're a sales manager and you need to prepare a month-end sales report to keep your boss happy. If the task is straightforward, you can develop a VBA program to do it for you. Your boss will be impressed by the consistently high quality of your reports, and you'll be promoted to a new job for which you are highly unqualified.

Automating repetitive operations

If you need to perform the same action on, say, 12 different Excel workbooks, you can record a macro while you perform the task on the first workbook and then let the macro repeat your action on the other workbooks. The nice thing about this is that Excel never complains about being bored. Excel's macro recorder is similar to recording sound on a tape recorder. But it doesn't require a microphone.

Creating a custom command

Do you often issue the same sequence of Excel menu commands? If so, save yourself a few seconds by developing a macro that combines these commands into a single custom command, which you can execute with a single keystroke or button click.

Creating a custom button

You can customize your Quick Access Toolbar with your own buttons that execute the macros you write. Office workers tend to be very impressed by this sort of thing.

Developing new worksheet functions

Although Excel includes numerous built-in functions (such as SUM and AVERAGE), you can create *custom* worksheet functions that can greatly simplify your formulas. I guarantee you'll be surprised by how easy this is. (I show you how to do this in Chapter 21.) Even better, the Insert Function dialog box displays your custom functions, making them appear built in. Very snazzy stuff.

Creating complete, macro-driven applications

If you're willing to spend some time, you can use VBA to create large-scale applications complete with a custom Ribbon, dialog boxes, on-screen help, and lots of other accoutrements. This book doesn't go quite that far, but I'm just telling you this to impress you with how powerful VBA really is.

Creating custom add-ins for Excel

You're probably familiar with some of the add-ins that ship with Excel. For example, the Analysis ToolPak is a popular add-in. You can use VBA to develop your own special-purpose add-ins. I developed my Power Utility Pak add-in by using only VBA, and people all around the world use it.

Advantages and Disadvantages of VBA

In this section, I briefly describe the good things about VBA — and I also explore its darker side.

VBA advantages

You can automate almost anything you do in Excel. To do so, you write instructions that Excel carries out. Automating a task by using VBA offers several advantages:

- Excel always executes the task in exactly the same way. (In most cases, consistency is a good thing.)
- Excel performs the task much faster than you can do it manually (unless, of course, you're Clark Kent).
- If you're a good macro programmer, Excel always performs the task without errors (which probably can't be said about you or me).
- If you set things up properly, someone who doesn't know anything about Excel can perform the task.
- You can do things in Excel that are otherwise impossible — which can make you a very popular person around the office.
- For long, time-consuming tasks, you don't have to sit in front of your computer and get bored. Excel does the work, while you hang out at the water cooler.

VBA disadvantages

It's only fair that I give equal time to listing the disadvantages (or *potential* disadvantages) of VBA:

- You have to find out how to write programs in VBA (but that's why you bought this book, right?). Fortunately, it's not as difficult as you might expect.
- Other people who need to use your VBA programs must have their own copies of Excel. It would be nice if you could press a button that transforms your Excel/VBA application into a stand-alone program, but that isn't possible (and probably never will be).
- Sometimes, things go wrong. In other words, you can't blindly assume that your VBA program will always work correctly under all circumstances. Welcome to the world of debugging and, if others are using your macros, technical support.
- VBA is a moving target. As you know, Microsoft is continually upgrading Excel. Even though Microsoft puts great effort into compatibility between versions, you may discover that VBA code you've written for Excel 2007 doesn't work properly with older versions or with a future version of Excel.

VBA in a Nutshell

Just to let you know what you're in for, I've prepared a quick and dirty summary of what VBA is all about. Of course, I describe all this stuff in semi-excruciating detail later in the book.

- **You perform actions in VBA by writing (or recording) code in a VBA module.** You view and edit VBA modules by using the Visual Basic Editor (VBE).
- **A VBA module consists of Sub procedures.** A Sub procedure has nothing to do with underwater vessels or tasty sandwiches. Rather, it's computer code that performs some action on or with objects (discussed in a moment). The following example shows a simple Sub procedure called AddEmUp. This amazing program displays the result of 1 plus 1.

```
Sub AddEmUp()
    Sum = 1 + 1
    MsgBox "The answer is " & Sum
End Sub
```

- **A VBA module can also have Function procedures.** A Function procedure returns a single value. You can call it from another VBA procedure or even use it as a function in a worksheet formula. An example of a Function procedure (named AddTwo) follows. This Function accepts two numbers (called arguments) and returns the sum of those values.

```
Function AddTwo(arg1, arg2)
    AddTwo = arg1 + arg2
End Function
```

- **VBA manipulates objects.** Excel provides dozens and dozens of objects that you can manipulate. Examples of objects include a workbook, a worksheet, a cell range, a chart, and a Shape. You have many more objects at your disposal, and you can manipulate them by using VBA code.
- **Objects are arranged in a hierarchy.** Objects can act as *containers* for other objects. At the top of the object hierarchy is Excel. Excel itself is an object called Application. The Application object contains other objects such as Workbook objects and Add-In objects. The Workbook object can contain other objects, such as Worksheet objects and Chart objects. A Worksheet object can contain objects such as Range objects and PivotTable objects. The term *object model* refers to the arrangement of these objects. (Object model mavens can find out more in Chapter 4.)
- **Objects of the same type form a collection.** For example, the Worksheets collection consists of all the worksheets in a particular workbook. The Charts collection consists of all Chart objects in a workbook. Collections are themselves objects.

- **You refer to an object by specifying its position in the object hierarchy, using a dot (that is, a period) as a separator.** For example, you can refer to the workbook Book1.xlsx as

```
Application.Workbooks("Book1.xlsx")
```

  This refers to the workbook Book1.xlsx in the Workbooks collection. The Workbooks collection is contained in the Application object (that is, Excel). Extending this to another level, you can refer to Sheet1 in Book1.xlsx as

```
Application.Workbooks("Book1.xlsx").Worksheets("Sheet1
        ")
```

  As shown in the following example, you can take this to still another level and refer to a specific cell (in this case, cell A1):

```
Application.Workbooks("Book1.xlsx").Worksheets("Sheet1
        ").Range("A1")
```

- **If you omit specific references, Excel uses the *active* objects.** If Book1.xlsx is the active workbook, you can simplify the preceding reference as follows:

```
Worksheets("Sheet1").Range("A1")
```

  If you know that Sheet1 is the active sheet, you can simplify the reference even more:

```
Range("A1")
```

- **Objects have properties.** You can think of a property as a *setting* for an object. For example, a Range object has such properties as Value and Address. A Chart object has such properties as HasTitle and Type. You can use VBA to determine object properties and also to change properties.
- **You refer to a property of an object by combining the object name with the property name, separated by a dot.** For example, you can refer to the Value property in cell A1 on Sheet1 as follows:

```
Worksheets("Sheet1").Range("A1").Value
```

- **You can assign values to variables.** A *variable* is a named element that stores information. You can use variables in your VBA code to store such things as values, text, or property settings. To assign the value in cell A1 on Sheet1 to a variable called *Interest,* use the following VBA statement:

```
Interest = Worksheets("Sheet1").Range("A1").Value
```

- **Objects have methods.** A *method* is an action Excel performs with an object. For example, one of the methods for a Range object is ClearContents. This method clears the contents of the range.
- **You specify a method by combining the object with the method, separated by a dot.** For example, the following statement clears the contents of cell A1:

```
Worksheets("Sheet1").Range("A1").ClearContents
```

- **VBA includes all the constructs of modern programming languages, including arrays and looping.** In other words, if you're willing to spend a little time mastering the ropes, you can write code that does some incredible things.

Believe it or not, the preceding list pretty much describes VBA in a nutshell. Now you just have to find out the details. That's the purpose of the rest of this book.

An Excursion into Versions

If you plan to develop VBA macros, you should have some understanding of Excel's history. I know you weren't expecting a history lesson when you picked up this book, but bear with me. This is important stuff.

Here are all the major Excel for Windows versions that have seen the light of day, along with a few words about how they handle macros:

- **Excel 2:** The original version of Excel for Windows was called Version 2 (rather than 1) so that it would correspond to the Macintosh version. Excel 2 first appeared in 1987 and nobody uses it anymore, so you can pretty much forget that it ever existed.
- **Excel 3:** Released in late 1990, this version features the XLM macro language. Nobody uses this version either.
- **Excel 4:** This version hit the streets in early 1992. It also uses the XLM macro language. A small number of people still use this version. (They subscribe to the philosophy *if it ain't broke, don't fix it*.)
- **Excel 5:** This one came out in early 1994. It was the first version to use VBA (but it also supports XLM). Excel 5 users are becoming increasingly rare.

- **Excel 95:** Technically known as Excel 7 (there is no Excel 6), this version began shipping in the summer of 1995. It's a 32-bit version and requires Windows 95 or Windows NT. It has a few VBA enhancements, and it supports the XLM language. Occasionally, I'll run into someone who still uses this version.
- **Excel 97:** This version (also known as Excel 8) was born in January, 1997. It has *many* enhancements and features an entirely new interface for programming VBA macros. Excel 97 also uses a new file format (which previous Excel versions cannot open). A fair number of people continue to use this version.
- **Excel 2000:** This version's numbering scheme jumped to four digits. Excel 2000 (also known as Excel 9) made its public debut in June 1999. It includes only a few enhancements from a programmer's perspective, with most enhancements being for users — particularly online users. With Excel 2000 came the option to digitally sign macros, thus enabling you to guarantee your users that the code delivered is truly yours. Excel 2000 still has a modest number of users.
- **Excel 2002:** This version (also known as Excel 10 or Excel XP) appeared in late 2001. Perhaps this version's most significant feature is the ability to recover your work when Excel crashes. This is also the first version to use copy protection (known as *product activation*).
- **Excel 2003:** Of all the Excel upgrades I've ever seen (and I've seen them all), Excel 2003 has the fewest new features. In other words, most hard-core Excel users (including yours truly) were very disappointed with Excel 2003. Yet people still bought it. I think these were the folks moving up from a pre-Excel 2002 version.
- **Excel 2007:** The latest, and without a doubt, the greatest. Microsoft outdid its corporate self with this version. Excel 2007 has a new look, a new user interface, and now supports more than a million rows. This book is written for Excel 2007, so if you don't have this version, you're reading the wrong book.

So what's the point of this mini history lesson? If you plan to distribute your Excel/VBA files to other users, it's vitally important that you understand which version of Excel they use. People using an older version won't be able to take advantage of features introduced in later versions. For example, if you write VBA code that references cell XFD1048576 (the last cell in a workbook) , those who use an earlier version will get an error because pre-Excel 2007 worksheets only had 65,536 rows and 255 columns (the last cell is IV65536). Excel 2007 also has some new objects, methods, and properties. If you use these in your code, users with an older version of Excel will get an error when they run your macro — and you'll get the blame.