INTRODUCTION

Visual Basic for Applications is a mouthful to say (which is why I'll use the standard short form—VBA—from now on), but it also seems like it would be a real handful to learn. After all, this is a *programming language* we're talking about, right?

True, but VBA was designed to be easy to learn and straightforward to apply. I've learned a couple of dozen programming languages over the past 30 years or so, and I can tell you that VBA is, hands down, the easiest language I've ever worked with.

Okay, but isn't this stuff just for power users and the staff of the Information Technology department?

Yes, VBA is a useful tool for hardcore users and those who need to design major projects. But VBA can be immensely useful for *every* user. As a writer, I use Word constantly, and over the years I've developed dozens of small macros, functions, and forms that streamline or automate repetitive chores. Most of these routines consist of only a few lines of code, and each one saves me only about 30 seconds to a minute, depending on the task. But I use these routines 50 or 100 times a day, so I end up saving myself anywhere from 30 to 90 minutes a day! That's pretty remarkable, but the proof is in the pudding: I can now write far more pages in a day than I used to. (Don't tell my editor!)

Whether your concern is ease-of-use or personal productivity, there's little doubt VBA can make working with the Office applications a better experience. So now all you have to do is learn how to use it, and that's where this book comes in. My goal in writing this book was to give you an introduction to the VBA language, and to give you plenty of examples for putting the language to good use. Even if you've never even programmed your VCR, this book will teach you VBA programming from the ground up. The first six chapters, in particular, give you all the know-how you'll need to be a competent and productive programmer.

What Is a Macro?

It doesn't matter in which Office program you're working—it could be Word, it could be Excel, it could be PowerPoint. A few times a day you probably find yourself performing some chore that either you've done dozens of times in the past, or that you have to repeat a number of times in a row. It could be typing and formatting a section of text, running a series of menu commands, or editing a document in a particular way. If you're like most people, when faced with these repetitive chores, you probably find yourself wishing there was some way to ease the drudgery and reduce the time taken by this mindless but necessary work.

Sure, most of the Office applications have a Repeat button on the Quick Access Toolbar that lets you repeat your most recent action. That's handy, but it repeats only a single action. If you need to repeat two or more actions, this solution doesn't work.

What's a person to do about this? Well, what if I told you that it was possible to *automate* just about any routine and repetitive task? What if I told you that it was possible to take this automated task and run it immediately simply by selecting a command or even by just pressing a key or clicking a button?

It sounds too good to be true, I know, but that's just what Visual Basic for Applications (VBA) can do for you. You use VBA to create something called a *macro*, which is really just a series of tasks that you want a program to perform. So a macro is not unlike a recipe, which is a set of instructions that tells you what tasks to perform to cook or bake something. A macro, too, is a set of instructions, but in this case it tells a program (such as Word or Excel) what tasks to perform to accomplish some goal.

The big difference, however, is that a macro combines all these instructions into a single script that you can invoke with a keystroke or just a few mouse clicks. In this sense, then, a macro isn't so much like a recipe for, say, how to bake bread, but is more akin to a bread machine, which, after it has been loaded with ingredients, bakes a loaf with the push of a button.

This list of instructions is composed mostly of *macro statements*. Some of these statements perform specific macro-related tasks, but most correspond to the underlying application's commands and dialog box options. For example, in any application, you can close the current (active) window by selecting the Office menu's Close command. In a VBA macro, the following statement does the same thing:

```
ActiveWindow.Close
```

What Does VBA Have to Do with Macros?

VBA is a programming language designed specifically for creating application macros. That sounds intimidating, I'm sure, but VBA's biggest advantage is that it's just plain easier to use than most programming languages. If you don't want to do any programming, VBA enables you to record macros and attach them to buttons either inside a document or on the Quick Access Toolbar (as you'll see in Chapter 1). You also can create your own dialog boxes by "drawing" the appropriate controls onto a document. Other visual tools enable you to customize the Ribbon as well, so you have everything you need to create simple scripts without writing a line of code.

Of course, if you want to truly unleash VBA's capabilities, you'll need to augment your interface with programming code. That sounds pretty fancy, but the VBA language is constructed in such a way that it's fairly easy to get started and to figure things out as you go along. More than any other programming language, VBA enables you to do productive things without climbing a huge learning curve.

What You Should Know Before Reading This Book

First and foremost, this book does *not* assume that you've programmed before. VBA beginners are welcome here and will find the text to their liking.

I've tried to keep the chapters focused on the topic at hand and unburdened with long-winded theoretical discussions. For the most part, each chapter gets right down to brass tacks without much fuss and bother. To keep the chapters uncluttered, I've made a few assumptions about what you know and don't know:

- I assume you have knowledge of rudimentary computer concepts such as files and folders.
- I assume you're familiar with Windows and that you know how to launch applications and work with tools such as menus, dialog boxes, and the Help system.
- I assume you can operate peripherals attached to your computer, such as the keyboard, mouse, printer, and modem.
- This book's examples use the Office 2007 applications, although most of them also work with Office 2000, Office XP, and Office 2003. Therefore, I assume you've used these Office programs for a while and are comfortable working with these programs.

What's in the Book

This book isn't meant to be read from cover to cover, although you're certainly free to do just that if the mood strikes you. Instead, most of the chapters are set up as self-contained units that you can dip into at will to extract whatever nuggets of information you need. However, if you're a beginning VBA programmer, I recommend working through Chapters 1 to 6 to ensure that you have a thorough grounding in the fundamentals of the VBA editor and the VBA language.

The book is divided into four main parts. To give you the big picture before diving in, here's a summary of what you'll find in each part:

> **Part 1, "Getting Started with VBA"**—The half dozen chapters in Part 1 give you a thorough grounding in VBA. You start off easy by learning how to create and run recorded macros, which doesn't require any programming (Chapter 1). In Chapters 2 though 6, you learn the basics of the VBA language, which you'll use in earnest throughout the rest of the book.

> **Part 2, "Putting VBA to Work"**—The five chapters in Part 2 enable you to put your VBA programming knowledge to good and practical use by showing you how to program the five main Office applications: Word (Chapter 7), Excel (Chapter 8), PowerPoint (Chapter 9), Access (Chapter 10), and Outlook (Chapter 11).

> **Part 3, "Getting the Most Out of VBA"**—The five chapters in Part 3 augment your VBA toolkit with lots of useful techniques. You learn how to interact with users (Chapter 12), how to create custom dialog boxes (Chapter 13), how to create custom Ribbon tabs and buttons (Chapter 14), how to debug your VBA code (Chapter 15), and how to use VBA to control other applications (Chapter 16).

> **Appendixes**—The book finishes with two appendixes that you can use as a reference. Appendix A lists all the VBA statements, and Appendix B lists all the VBA functions.

This Book's Special Features

VBA for the 2007 Microsoft Office System is designed to give you the information you need without making you wade through ponderous explanations and interminable technical background. To make your life easier, this book includes various features and conventions that help you get the most out of the book and VBA itself.

- **Things you type:** Whenever I suggest that you type something, what you type appears in a **bold** font.

- **Commands:** I use the following style for Ribbon commands: View, Macros. This means that you click the Ribbon's View tab, and then click the Macros button.

- **Dialog box controls:** Dialog box controls have underlined accelerator keys: <u>C</u>lose.

- **Visual Basic keywords:** Keywords reserved in the VBA language appear in `monospace` type.

- **Code-continuation character (➥):** When a statement is too long to fit on one line of this book, I break it at a convenient place, and add the code-continuation character at the beginning of the next line.

This book also uses the following boxes to draw your attention to important (or merely interesting) information.

> **NOTE**
> The Note box presents asides that give you more information about the topic under discussion. These tidbits provide extra insights that give you a better understanding of the task at hand.

> **TIP**
> The Tip box tells you about VBA methods that are easier, faster, or more efficient than the standard methods.

> **CAUTION**
> The all-important Caution box tells you about potential accidents waiting to happen. There are always ways to mess things up when you're working with computers. These boxes help you avoid at least some of the pitfalls.

→ These cross-reference elements point you to related material elsewhere in the book.