



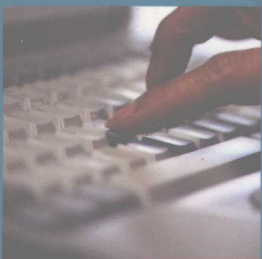
 **plus**
SERIES

NOT FOR RESALE
DONATION 02AS107



MICROSOFT®
Visual Basic
for Applications

Alan I Rea



**Mc
Graw
Hill** **Technology
Education**

+Plus Series

Microsoft® Visual Basic for Applications

Alan I Rea

Haworth College of Business

Western Michigan University



Boston Burr Ridge, IL Dubuque, IA Madison, WI New York San Francisco St. Louis
Bangkok Bogotá Caracas Kuala Lumpur Lisbon London Madrid Mexico City
Milan Montreal New Delhi Santiago Seoul Singapore Sydney Taipei Toronto



THE +PLUS SERIES: MICROSOFT® VISUAL BASIC FOR APPLICATIONS

Published by McGraw-Hill Technology Education, a business unit of The McGraw-Hill Companies, Inc., 1221 Avenue of the Americas, New York, NY, 10020. Copyright © 2005 by The McGraw-Hill Companies, Inc. All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system, without the prior written consent of The McGraw-Hill Companies, Inc., including, but not limited to, in any network or other electronic storage or transmission, or broadcast for distance learning.

Some ancillaries, including electronic and print components, may not be available to customers outside the United States.

This book is printed on acid-free paper.

1 2 3 4 5 6 7 8 9 0 QPD/QPD 0 9 8 7 6 5 4

ISBN 0-07-283616-4

Editor-in-chief: *Bob Woodbury*

Publisher: *Brandon Nordin*

Senior sponsoring editor: *Donald J. Hull*

Editorial assistant: *Alaina Grayson*

Manager, Marketing and Sales: *Paul Murphy*

Project manager: *Jim Labeots*

Senior production supervisor: *Rose Hepburn*

Design coordinator: *Cara David*

Senior digital content specialist: *Brian Nacik*

Cover design: *Cara David*

Typeface: *10.5/12 Minion*

Compositor: *PreMediaOne*

Printer: *Quebecor World Dubuque Inc.*

Library of Congress Cataloging-in-Publication Data

Rea, Alan I

Microsoft® Visual basic for applications 2003 / Alan Rea.

p. cm.

Includes index.

ISBN 0-07-283616-4 (alk. paper)

1. Microsoft® Visual Basic for applications. 2. BASIC (Computer program language) 3.

Microsoft® Windows (Compute file) I. Title.

QA76.73.B3R418 2005

005.26'8—dc22

2004053086

www.mhhe.com

dedication

To my wife, Lynda. Without you I couldn't do any of this. Thanks for listening to me ponder the merits of one programming structure over the other at the kitchen table even though I might as well be speaking in another language. I appreciate the smile and occasional nod.

The +Plus Series Approach

Students will be able to discover and apply learned skills while working through a variety of tasks. The +Plus Series textbooks are designed for engaged, active learning.

HOW WILL THE +PLUS SERIES ACCOMPLISH THIS?

- Through relevant, real-world scenarios and cases
- Through tasks in each chapter that incorporate steps and tips for easy reference
- By providing alternate ways and styles encouraging student involvement
- Through the end-of-chapter reinforcement projects that support what the student has learned

TO DATE +PLUS SERIES TITLES INCLUDE:

- Microsoft® FrontPage 2003 by Ann Willer
- Microsoft® Outlook 2003 by Brenda Nielsen
- Microsoft® Visual Basic for Applications (VBA) by Alan I Rea
- What's New in Microsoft® Office 2003 by Don Amoroso, Cheryl Manning, and Catherine Manning Swinson

+PLUS SERIES GOALS AND APPROACH TO LEARNING

The +Plus Series textbooks emphasize that students learn by being actively involved—by *doing*. Teaching how to accomplish tasks is not enough for thorough conceptual understanding. Students need to understand and be able to complete the presented tasks in order to master skills. The +Plus Series books are subdivided into sessions that contain groups of tasks and hands-on practice. The session tasks contain concept explanations combined with practice steps to apply the presented material.

VBA Key Features

Visual Basic for Applications empowers Microsoft Office users to develop their own customized applications to solve problems. At the beginning of each chapter, the text presents students with a **Chapter Case** that involves solving a business problem by developing a VBA macro. Whether students create a prototype for an e-commerce application or an automated report generator, they apply the programming concepts presented in the chapter through a series of guided tasks to create a working VBA enhanced Microsoft Office application.

As students work through the text they're exposed to programming structures and logic, as well as, VBA syntax. However, students always practice each programming concept before moving forward. It's not enough to know what programming controls structures are. Students must also know how and when to use one to solve a problem.

Students check their understanding through a **Task Reference**. **Task References** are interspersed at key learning points throughout the chapter. The **Task Reference** guides students through programming steps that build on each other as students create an application or practice a concept. At the end of each chapter, students can review each **Task Reference** to make sure they understand what they've applied.

Students also learn that there's more than one way to apply VBA to a problem. Throughout the chapters, **Another Way** illustrates a different method or shortcut to apply VBA. **Another Word** illustrates how programmers use different terms and approaches to the same problem.

Students can also test their comprehension after each section of a chapter with short fill-in-the-blank sentences in **Making the Grade**. These section checkpoints enable students to judge whether they comprehend the major concepts in each section before moving forward.

At the end of each chapter, the text presents students with three levels of concepts and programming exercises to reinforce what they've learned in the chapter.

Level One: Task Reference Roundup reviews the tasks students completed in the chapter. It lists chapter page numbers for reference as well as the major steps used to complete each task.

Level Two: Review of Concepts allows students to test their comprehension of key concepts in each chapter as well as apply the knowledge to different scenarios. **Level Two** consists of **Fill-In**, **Review Questions**, and a **Create the Question** exercise that asks students to provide the correct question when given an answer.

Level Three: Hands-On Projects builds on the knowledge and skills students have acquired in the chapter. Each **Level Three** consists of a set of programming exercises that allow students to practice and expand their programming knowledge and skills. **Level Three** consists of three levels of programming exercises: **Practice**, **Challenge!**, and a **Running Project**.

In the **Practice** section, students follow a set of steps to apply VBA skills learned in the chapter. These exercises show students how they can use the same techniques on different problems.

In the **Challenge!** Section, students must apply newly-acquired programming knowledge and VBA skills to a problem. The problems encourage students to implement what they have recently learned. While students solutions will vary, the exercises provide hints and final application screens to help students as they develop their application.

In the **Running Project**, students take on the role of an Information Technology staff person in a small business, La Llama Cycle. Students join the owner, Jesus Rodriguez, as he starts his customized motorcycle shop. They help him and the staff improve business processes using customized VBA solutions.

Alan I Rea is an associate professor of Computer Information Systems at Western Michigan University's Haworth College of Business. Alan holds a BA from The Pennsylvania State University, an MA from Youngstown State University, and a PhD from Bowling Green State University.

Alan teaches courses in various programming languages, Web and e-commerce application development, and system administration. He has published in various journals, including the *Journal of Information Systems Education*, the *Journal of Computer Information*

Systems, and the *Mid-American Journal of Business*. He regularly serves as a reviewer for conferences, such as the Americas Conference on Information Systems. Alan is a member of various professional committees concerned with teaching technology.

When not teaching or writing, Alan spends time playing the tuba, computer gaming, or hanging out with his family. Alan lives in Kalamazoo, Michigan, with his wife, Lynda, son, Aidan, two cats, and various forms of wildlife that include a growing roost of neighborhood turkey vultures.

Acknowledgments

Each book project is a different experience for me. While I've only been writing textbooks for about five years, I'm still amazed at the intensity of the process and the euphoria that comes with producing a high-quality product I know will make a difference in the classroom.

Don't think that a textbook such as this one is a solo effort. Without the professionalism and hard work on the project team, my words would never appear anywhere but on my computer screen and a few discarded print outs. I can't thank everyone in the process, but please take the time to look over the team list in this preface. Each and every one of these people make it their job to produce a quality product (and make me look good in the process).

I'd like especially to thank Don Hull, my senior sponsoring editor. Although Don came into the process after we started the text, it was his wonderful

rapport and calm demeanor that kept me on track. (The occasional e-mail and phone call helped as well.)

Of course, thanks go out to my colleagues and the administration at Western Michigan University. They've created an environment that encourages a student-centered approach to teaching which allows me to embark on adventures such as writing this textbook.

I'd also like to give my greatest thanks to my wife, Lynda. She has been my supporter, critic, and frontline editor. Her contributions to this textbook are too numerous to list. Without her, I never would've made it this far.

Finally, I want to thank my son, Aidan. Although he's new to the world he brings much joy to my life. He's also the person who keeps me rooted in reality with the occasional tap at the door asking "Dada" to come out and play.

table of contents

1

CHAPTER 1 VBA AND YOU: HOW CAN YOU GET THE MOST OUT OF MICROSOFT OFFICE?

1.1

CHAPTER CASE

DAGGITT DEVELOPMENT 1.2

SESSION 1.1

INTRODUCTION TO VBA PROGRAMMING 1.3

Where Did VBA Come From? 1.3

The Birth of VBA 1.5

General Programming Concepts 1.5

VBA's Role in Microsoft Office 1.7

SESSION 1.2

USING THE VBA INTEGRATED DEVELOPMENT

ENVIRONMENT (IDE) 1.8

Securing Microsoft Office 1.8

Opening the VBA IDE 1.10

Touring the VBA IDE 1.11

Project Explorer 1.12

Properties Window 1.14

Code Window 1.17

Customizing the VBA IDE Windows 1.18

Exploring the Excel VBA IDE 1.20

Comparing VBA IDEs 1.21

SESSION 1.3

RECORDING AND MANIPULATING MACROS 1.23

Recording Macros 1.23

Recording the Macro 1.24

Running a VBA Macro 1.26

Manipulating Macros 1.28

Changing the Code 1.29

Assigning Macros 1.31

Printing Macros 1.33

SESSION 1.4

SUMMARY 1.35

LEVEL ONE:

TASK REFERENCE ROUNDUP 1.37

LEVEL TWO:

REVIEW OF CONCEPTS 1.38

LEVEL THREE:

HANDS-ON PROJECTS 1.39

Practice 1.39

Challenge! 1.41

RUNNING PROJECT 1.43

2

CHAPTER 2 OBJECTS, VARIABLES, AND FUNCTIONS: HOW DO YOU ACCESS AND USE INFORMATION IN VBA?

2.1

CHAPTER CASE

NORTH AVENUE RACQUETBALL
FOUNDATION 2.2

SESSION 2.1

UNDERSTANDING AND IMPLEMENTING
OBJECTS 2.5

Understanding the Object Model 2.5

Using the Object Browser 2.10

Understanding Objects 2.13

Accessing and Using Objects 2.14

SESSION 2.2

UNDERSTANDING VARIABLES 2.19

Storing Data 2.19

Creating Variables 2.19

Assigning Data Types 2.21

Naming Variables 2.23

Managing Variables 2.23

Using an Object Variable 2.26

SESSION 2.3

USING VARIABLES AND FUNCTIONS 2.29

Using String Variables 2.29

Using the InputBox Function 2.31

Using Date and Time Variables 2.34

Using Number Variables 2.37

SESSION 2.4

SUMMARY 2.42

LEVEL ONE:

TASK REFERENCE ROUNDUP 2.43

LEVEL TWO:

REVIEW OF CONCEPTS 2.44

LEVEL THREE:

HANDS-ON PROJECTS 2.45

Practice 2.45

Challenge! 2.48

RUNNING PROJECT 2.50

3

CHAPTER 3 CONTROL STRUCTURES: HOW CAN YOU MAKE DECISIONS IN VBA?

3.1

CHAPTER CASE

TRI-CECTICKET RESERVATION SYSTEM
(TRICECTRS)

3.2

SESSION 3.1

CONTROL STRUCTURE OVERVIEW

3.5

Sequence Control Structure

3.5

Selection Control Structure

3.5

Repetition Control Structure

3.7

SESSION 3.2

IFTHEN ELSE STATEMENTS

3.8

Comparison Operators

3.9

Logical Operators

3.10

Implementing the IfThen Else Statement

3.10

Defensive Programming

3.17

Implementing Nested IfThen Else Statements

3.20

Implementing the MsgBox Function

3.21

SESSION 3.3

SELECT CASE STATEMENTS

3.26

Select Case Syntax

3.26

Implementing Select Case

3.27

SESSION 3.4

ITERATIONS AND LOOPS

3.30

Do Loops

3.30

Implementing a Do Loop

3.30

For Next Loops

3.34

Debugging Code during RunTime

3.38

SESSION 3.5

SUMMARY

3.41

LEVEL ONE:

TASK REFERENCE ROUNDUP

3.43

LEVEL TWO:

REVIEW OF CONCEPTS

3.44

LEVEL THREE:

HANDS-ON PROJECTS

3.45

Practice

3.45

Challenge!

3.50

RUNNING PROJECT

3.52

4

CHAPTER 4 APPLICATION DEVELOPMENT: HOW CAN YOU CREATE YOUR OWN MICROSOFT OFFICE APPLICATION?

4.1

CHAPTER CASE

Newton Sales International

4.2

SESSION 4.1

COMMUNICATING WITH USERS

4.5

Dialogs Collection

4.5

Microsoft Office Assistant

4.9

SESSION 4.2

CUSTOMIZING YOUR APPLICATION

4.17

Creating a Customized Dialog Box with Forms

4.17

Customizing the Newton Sales Letter

Dialog Box

4.26

SESSION 4.3

INTEGRATING OFFICE APPLICATIONS

4.37

Introducing the Basic Automatic Steps

4.37

Creating the Newton Sales Integrated

Application

4.43

Debugging and ErrorTrapping

4.54

SESSION 4.4

SUMMARY

4.57

LEVEL ONE:

TASK REFERENCE ROUNDUP

4.59

LEVEL TWO:

REVIEW OF CONCEPTS

4.60

LEVEL THREE:

HANDS-ON PROJECTS

4.61

Practice

4.61

Challenge!

4.66

RUNNING PROJECT

4.68

END OF BOOK

INDEX

EOB I.1

VBA and You:

How Can You Get the Most Out of Microsoft Office?

Chapter Objectives

- Learn VBA history
- Understand why to use VBA
- Understand programming concepts
- Use the Word VBA IDE
- Use the Excel VBA IDE
- Record macros
- Manipulate macros
- Implement macros
- Create programming solutions using VBA

chapter case

Daggitt Development

During her summer internship, Maggie, a junior at the university, wants to learn more about using computer applications in business. Luckily, she gets an internship at Daggitt Development (DD for short), a local consulting company that develops software applications for small businesses. DD provides customized software to businesses for new payroll applications, electronic appointment books, or any other type of office tool.

Cody, a senior programmer, tells Maggie that DD's most demanded service is the creation of macro programs for businesses to use within Microsoft Office. To do this, DD uses Visual Basic for Applications, or VBA. Using VBA, DD can tailor Microsoft Word, Excel, Access, PowerPoint, and Outlook to meet a customer's needs. Cody notes that DD transforms aspects of horizontal software into vertical software. Although Maggie is not sure what this means, Cody assures her she will learn soon enough.

Because Maggie has experience with Microsoft Office applications, Cody is ready to start her on her first project. Maggie will develop a customized macro in Microsoft Office for a local business, Bob's Baklava. This macro is similar to the one DD uses to create the heading on its Weekly Development Project Report (see Figure 1.1). Cody tells her not to worry that she hasn't used VBA before. He knows that she can learn how to use VBA as she works on her project.

Maggie's first tasks are to learn about VBA and programming concepts, become familiar with the VBA Integrated Development Environment (IDE), and learn how to record, edit, and run VBA macros. This chapter will help both you and Maggie become better prepared to customize applications and get the most out of Microsoft Office.

Introduction

Chapter 1 introduces you to VBA programming and why people use it to customize applications. You'll also learn some of the history behind VBA and how to think and write like a programmer. We'll spend time touring the VBA IDE so that you are familiar with this programming environment. Finally, you'll learn how to record, manipulate, and implement VBA macros to customize Microsoft Word and Excel.

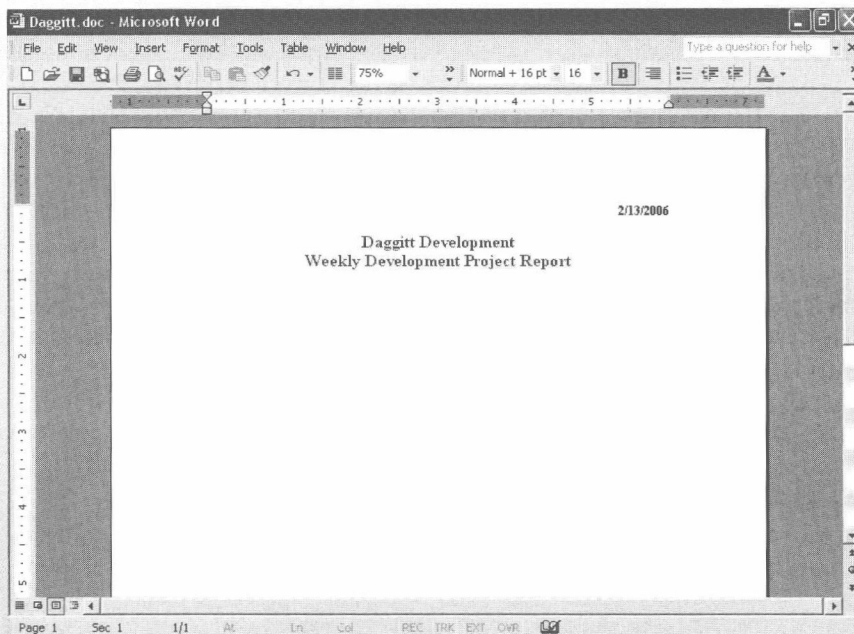


FIGURE 1.1

Daggitt Development's
Weekly Development
Project Report

SESSION 1.1 INTRODUCTION TO VBA PROGRAMMING

Like Maggie, you may think you don't have the necessary programming knowledge and skills to create customized Microsoft Office applications. However, many companies use VBA because it's easy to learn and use, yet powerful enough to solve complicated problems. **VBA**, or **Visual Basic for Applications**, is a programming language that works within certain software applications. A **programming language** contains specific rules and words that explain the logical steps to solve a problem. Although VBA works within many software applications, it's primarily used in Microsoft Office applications: Access, Excel, Outlook, PowerPoint, and Word. VBA is the programming language of choice for end user development of Microsoft Office applications. **End user development** is when computer users (such as you) develop and maintain computer applications with little or no help from technical specialists. However, you still need some programming knowledge and skills to work with VBA. The more programming knowledge and skills you have, the more you'll be able to accomplish using VBA.

In this section we'll explain where VBA came from, share some general characteristics of programming languages, and finally explain why we need VBA in Microsoft Office.

Where Did VBA Come From?

Before VBA was a programming language called BASIC. **BASIC** stands for Beginner's All-purpose Symbolic Instruction Code. Two professors from Dartmouth College—John Kemeny and Thomas Kurtz—created BASIC in 1964 so that students would have a simple language to learn how to program computers. BASIC's popularity grew and before long (in 1969) an eighth grader named Bill Gates started using it. Of course, this is the same Bill Gates who started a company called Micro-Soft with his friend Paul Allen in 1975. You can only imagine what happened after that.

BASIC Evolution

During the 1970s, BASIC took on various forms as Gates and Allen applied—or ported—it to many different computer systems, such as Altair, Apple, Commodore, and Atari. A computer language is **portable** when it has the ability to work on a variety

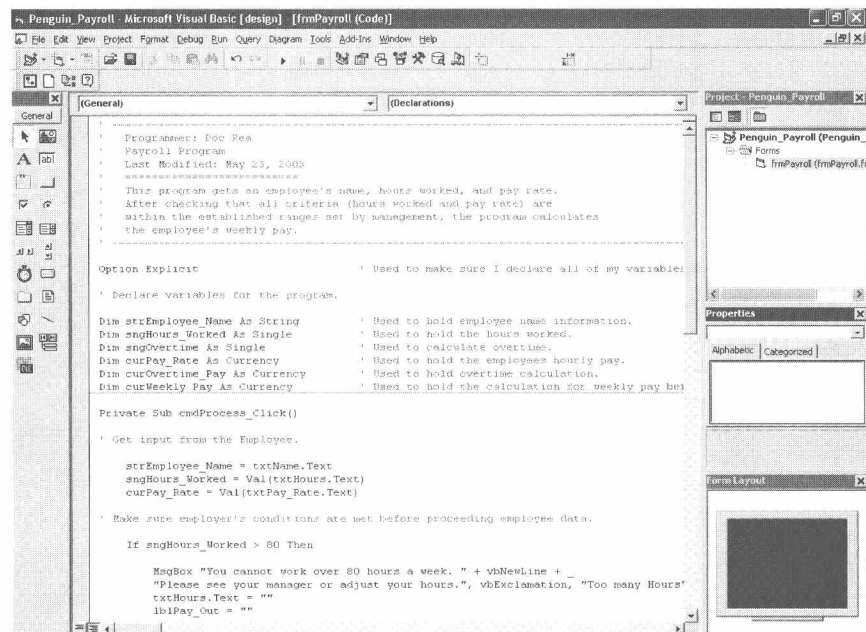
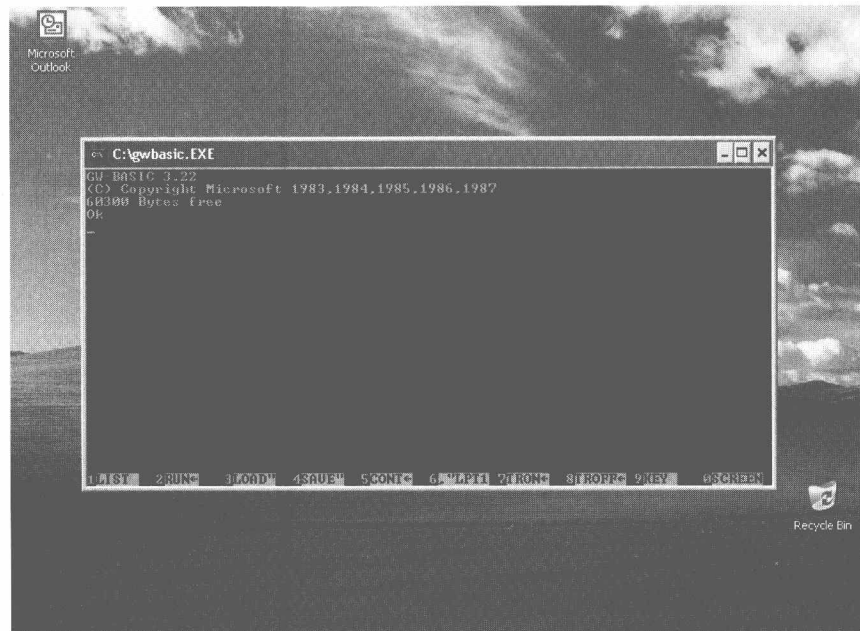
CHAPTER OUTLINE

- 1.1 Introduction to VBA Programming VBA 1.3**
- 1.2 Using the VBA Integrated Development Environment (IDE) VBA 1.8**
- 1.3 Recording and Manipulating Macros VBA 1.23**
- 1.4 Summary VBA 1.35**

of computers. Microsoft next developed an operating system called MS-DOS for IBM's first personal computer. It included a programming language called GW-BASIC. Microsoft was well on its way to becoming the powerful company it is today.

BASIC has many forms, but the most popular is probably Visual Basic. **Visual Basic** or **VB** is a graphical event-driven programming language. An **event-driven programming language** relies on actions and events to run. For example, an action or event occurs when you click on an icon or type a word. Visual Basic relies on a graphical user interface, such as Windows. A **graphical user interface**, or **GUI**, is a graphic- or icon-driven interface on which you point and click with your mouse to use software. Figure 1.2 is a comparison of the GW-BASIC programming environment with the VB integrated development environment (IDE). An **integrated development**

FIGURE 1.2
Comparison of GW-BASIC
and the VB IDE



environment, or *IDE*, is an application that provides programming tools to create, debug, and manage software programs. Notice how many options are available to the VB programmer as compared to the GW-BASIC programmer. More importantly, which one would you rather look at? Remember, these languages have the same source—BASIC—even though they're different.

The Birth of VBA

Microsoft introduced Visual Basic in 1992. Programmers use VB to create software programs. A *programmer* is a specialist who writes software to meet users' needs. Also in 1992, Microsoft released its first version of the popular Microsoft Office productivity suite. But it wasn't until 1997, when Microsoft released Office 97, that Microsoft included VBA in nearly all of the Office applications: Access, Excel, PowerPoint, and Word. Finally, in Office 2000, Microsoft included VBA in Outlook. Microsoft Office XP and Microsoft Office 2003 include even more VBA functionality.

What Is VBA?

When you program in Visual Basic, you're using a programming language to develop a complete software program that will run by itself. For example, you might create a game or accounting software. By contrast, when you use VBA, you're using a programming language that will create customized solutions within a software application, such as Microsoft Word. More specifically, VBA is a scripting language. A *scripting language* is a programming language that works within another application to perform tasks.

Maybe you want to allow users to customize how their Word documents will look. Or perhaps you want users to be able to track dinner reservations in an Excel spreadsheet without having to know how to use Excel. Or maybe you want to send formatted e-mail messages from a list of e-mail addresses in Access. With VBA, you can do all of this.

Why Use VBA?

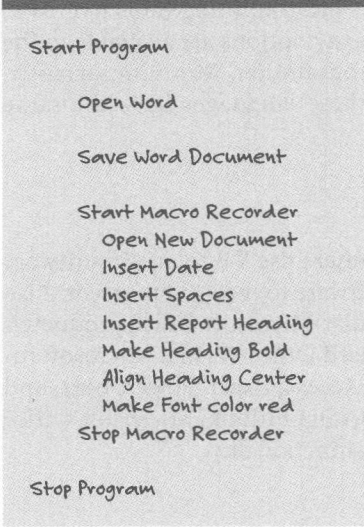
You know that most businesses use Microsoft Office because it meets many of their everyday business needs. Business users need to type letters and memos (Microsoft Word), manage budgets and figures (Microsoft Excel), send and receive e-mail (Microsoft Outlook), and give presentations (Microsoft PowerPoint). Some business users also need to keep track of inventories or other data (Microsoft Access). Microsoft Office can meet all of these needs for the majority of users because it's horizontal market software. *Horizontal market software* is general business software that has applications in many industries.

Although Microsoft Office does meet most business needs, a business often requires more specialized applications. One solution is to hire a consulting company to write a new software program. But many times a business can use VBA instead simply to customize Microsoft Office applications and solve the problem. VBA allows a business to create vertical market software out of a Microsoft Office application. *Vertical market software* is software that is unique to a particular industry. A business that uses VBA to customize an existing Microsoft Office application such as Excel, instead of hiring a consultant, saves a great deal of money.

General Programming Concepts

Now you know VBA's history and why you should use it. You probably are eager to learn how to use VBA effectively. But first you'll need to understand some basic programming concepts as well as how programmers solve business problems so you can understand how to apply VBA to a business requirement.

FIGURE 1.3
Maggie's pseudocode.



```
Start Program
Open Word
Save Word Document
Start Macro Recorder
Open New Document
Insert Date
Insert Spaces
Insert Report Heading
Make Heading Bold
Align Heading Center
Make Font color red
Stop Macro Recorder
Stop Program
```

Thinking Like a Programmer

As a programmer, your first priority is to know what a business needs. In this text, we'll help you pinpoint these needs. Figuring out a business problem takes practice, and you first must learn how to program to solve problems. Don't worry, though; we include some exercises that allow you to tackle this step on your own.

Once programmers understand the problem—such as a company's need for a payroll program to automatically print weekly paychecks—they map out the necessary steps to solve it. Programmers map out the problem in pseudocode. **Pseudocode** uses English statements to create an outline of the steps necessary for a piece of software to operate. Programmers call these steps an algorithm. An **algorithm** is a set of specific steps that solve a problem or carry out a task. An algorithm is like a

dessert recipe, in that a recipe lists all the steps necessary to create a scrumptious dessert. In programming, the sweet reward is a working piece of software.

You'll see examples of pseudocode throughout this text. We use it to describe each programming problem we approach. Figure 1.3 is an example of pseudocode Maggie wrote to help solve her first programming assignment in VBA. Notice that Maggie didn't type her pseudocode. Some programmers type pseudocode and some write it. We'll put this pseudocode to work for us later in the chapter.

Programmers also use program flowcharts to plot out the algorithm. A **program flowchart** is a graphical depiction of the detailed steps that a piece of software will perform. We'll use program flowcharts later in the text as we solve programming problems.

Once programmers write their algorithm in pseudocode or a program flowchart, they test it to make sure there are no logic errors. A **logic error** is a mistake in the way an algorithm solves a problem. For example, a payroll program is supposed to calculate overtime for anyone working more than 40 hours a week. If the program doesn't calculate overtime for someone working 50 hours a week, for example, it's a logic error.

Writing Like a Programmer

Now that you're familiar with how programmers think, let's look at how they write. Programmers call the process of writing software coding. **Coding** is when you translate your algorithm into a programming language. Coding looks different depending on what type of programming language you're using. This is because each programming language has a specific syntax. **Syntax** is a set of rules to follow. We'll focus only on VBA syntax in this text. Figure 1.4 shows an example of VBA code.

Notice that some words appear in blue. These are reserved words. **Reserved words** are words that a programming language has set aside for its own use. In Figure 1.4, you'll notice `Dim` is colored blue. `Dim` is a reserved word VBA uses to create a variable. We'll discuss variables in detail in Chapter 2. Notice also that many lines start with a single apostrophe and are a different color (green). Programmers call these explanations comments. **Comments** tell other programmers what's happening in software code. The computer ignores comment lines when it runs code.

tip: Although our figures are not in color, note these colors as you work in your code throughout the book.

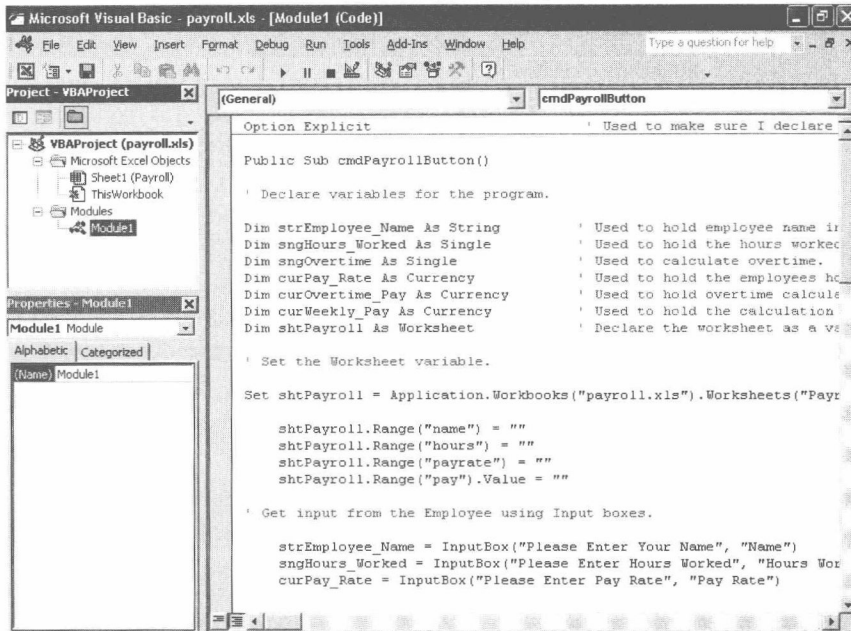


FIGURE 1.4

Notice how the VBA reserved words and comments are colored.

Working Like a Programmer

You'll spend most of this text learning how to code in VBA, so we'll save the discussion of how to code until later. After programmers have coded a program, they spend some time debugging their code. **Debugging** is the process of finding errors in software code. **Bugs** are a common name for software errors. When you debug your code, you look for syntax and run-time errors.

Syntax errors are mistakes in a software code's grammar. Just as misspelling a word is a mistake when writing, misspelling a word or forgetting to mark a comment correctly will cause a syntax error. If you're supposed to use a semicolon (;) but you use a colon (:) instead, you've made a syntax error. **Run-time errors** are mistakes that occur when you run the software code. Software not displaying a window correctly is a run-time error.

VBA's Role in Microsoft Office

You are an experienced Microsoft Office user. Using Office applications to solve business problems comes easily to you. Why then do you need VBA to help you with Microsoft Office? The answer is simple: you can do more in a shorter period of time.

We've discussed how VBA can make a horizontal market application into a vertical market application. In the next session, we'll show you how you can tailor your Office applications using the macro recorder. A **macro** is a scripting language program that performs a task or a series of tasks. However, using VBA you can go beyond macros and create VBA programs that add functionality and features to Office applications. In this text you'll learn how to create these VBA programs.

Ultimately with Microsoft Office and VBA you can achieve the following:

- **Use the existing power of Microsoft Office** When you begin with an already powerful suite of software applications, you have a distinct advantage over programmers who must program every part of their software application.
- **Add features and functionality to applications quickly** Using VBA you can customize an existing application or create a new application by combining existing Office applications. For example, you can create "Employee of the Month" certificates in Word using data stored in Excel.