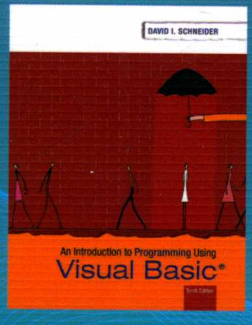
国外计算机科学教材系列

Pearson

精心改编

An Introduction to Programming Using Visual Basic, Tenth Edition



Visual Basic 程序设计教程

（第十版）（英文版）

［美］ David I. Schneider 著

罗 凌 改编

中国工信出版集团

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

国外计算机科学教材系列

Visual Basic 程序设计教程

（第十版）（英文版）

An Introduction to Programming Using Visual Basic

Tenth Edition

［美］David I. Schneider 著

罗 凌 改编

電子工業出版社
Publishing House of Electronics Industry
北京 • BEIJING

内容简介

David I. Schneider的Visual Basic教材自出版以来，深受读者欢迎，第十版对前几版的内容进行了全面的修订与更新。本书不是粗略地论及众多主题，而是针对重要问题进行深入分析。全书共10章，主要内容包括：Visual Basic简介、控件与事件，变量、输入和输出，分支结构，通用过程，循环结构，数组，其他控件和对象，面向对象编程，数据库，以及文本文件。全书给出了100多个示例和大量的习题，并且提供了实践性很强的程序设计项目，帮助读者掌握所学知识。

本书不仅可以作为Visual Basic程序设计课程的双语教材，也是广大计算机爱好者及程序开发人员学习Visual Basic的很好的参考用书。

版权贸易合同登记号 图字：01-2018-6073

图书在版编目（CIP）数据

Visual Basic程序设计教程：第十版：英文/（美）戴维·I. 施耐德（David I. Schneider）著；罗凌改编.
北京：电子工业出版社，2019.3
（国外计算机科学教材系列）
书名原文：An Introduction to Programming Using Visual Basic, Tenth Edition
ISBN 978-7-121-29543-0

Ⅰ.①V… Ⅱ.①戴… ②罗… Ⅲ.①BASIC语言－程序设计－教材－英文 Ⅳ.①TP312.8

中国版本图书馆CIP数据核字（2019）第030268号

策划编辑：冯小贝
责任编辑：冯小贝
印　　刷：三河市鑫金马印装有限公司
装　　订：三河市鑫金马印装有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路173信箱　邮编：100036
开　　本：787×1092　1/16　印张：35.5　字数：1181千字
版　　次：2012年9月第1版（原著第8版）
　　2019年3月第2版（原著第10版）
印　　次：2019年3月第1次印刷
定　　价：108.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。
质量投诉请发邮件至zlts@phei.com.cn，盗版侵权举报请发邮件至dbqq@phei.com.cn。
本书咨询联系方式：fengxiaobei@phei.com.cn。

PREFACE

Visual Basic has been a widely used programming language since its introduction in 1991. Its latest incarnation, Visual Basic 2015, brings continued refinement of the language. Visual Basic programmers are enthusiastically embracing the powerful capabilities of the language. Likewise, students learning their first programming language will find VB the ideal tool to understand the development of computer programs.

My objectives when writing this text were as follows:

1. *To develop focused chapters.* Rather than covering many topics superficially, I concentrate on important subjects and cover them thoroughly.
2. *To use examples and exercises with which students can relate, appreciate, and feel comfortable.* I frequently use real data. Examples do not have so many embellishments that students are distracted from the programming techniques illustrated.
3. *To produce compactly written text that students will find both readable and informative.* The main points of each topic are discussed first and then the peripheral details are presented as comments.
4. *To teach good programming practices that are in step with modern programming methodology.* Problem solving techniques and structured programming are discussed early and used throughout the book. The style follows object-oriented programming principles.
5. *To provide insights into the major applications of computers.*

What's New in the Tenth Edition

Among the changes in this edition, the following are the most significant.

1. ***Visual Basic Upgraded*** The version of Visual Basic has been upgraded from Visual Basic 2012 to Visual Basic 2015, and relevant new features of Visual Basic 2015 have been addressed.
2. ***Additional Exercises*** Sixty new exercises have been added, most of which are application exercises.
3. ***Updated Data*** The real-world data appearing in exercises, examples, and data files has been updated.
4. ***Decimal Data Type*** The Decimal data type has been introduced and used in all examples and exercises dealing with financial data.
5. ***Short-Circuit Evaluation*** AndAlso and OrElse are introduced for the evaluation of logical operators.
6. ***Windows 10*** The screen captures have been updated from Windows 8 to Windows 10 captures.
7. ***New Statements and Methods*** The Exit Sub and Exit Function statements and the string methods Remove and Replace are discussed.

Unique and Distinguishing Features

Exercises for Most Sections. Each section that teaches programming has an exercise set. The exercises both reinforce the understanding of the key ideas of the section and challenge the student to explore applications. Most of the exercise sets require the student to trace programs, find errors, and write programs. The answers to all the odd-numbered exercises in Chapters 1 through 6 and the short-answer odd-numbered exercises from Chapters 7, 8, 9 , and 10 are given at the end of the text. A screen capture accompanies most programming answers.

Practice Problems. Practice Problems are carefully selected exercises located at the end of a section, just before the exercise set. Complete solutions are given following the exercise set. The practice problems often focus on points that are potentially confusing or are best appreciated after the student has thought about them. The reader should seriously attempt the practice problems and study their solutions before moving on to the exercises.

Programming Projects. Beginning with Chapter 2, every chapter contains programming projects. The programming projects not only reflect the variety of ways that computers are used in the business community, but also present some games and general-interest topics. The large number and range of difficulty of the programming projects provide the flexibility to adapt the course to the interests and abilities of the students. Some programming projects in later chapters can be assigned as end-of-the-semester projects.

Comments. Extensions and fine points of new topics are deferred to the "Comments" portion at the end of each section so that they will not interfere with the flow of the presentation.

Captions. Every example and applied exercise is labeled with a caption identifying its type of application.

Screen Captures. The output for most applied exercises and programming projects are shown in screen captures. This feature helps clarify the intent of the exercises.

Case Studies. Each of the three case studies focuses on an important programming application. The problems are analyzed and the programs are developed with top-down charts and pseudocode. The programs can be downloaded from the companion website at http://www.pearsonhighered.com/schneider.

Chapter Summaries. In Chapters 1 through 10, the key results are stated and the important terms are summarized at the end of the chapter.

"How To" Appendix. Appendix B provides a compact, step-by-step reference on how to carry out standard tasks in the Visual Basic and Windows environments.

Appendix on Debugging. The placing of the discussion of Visual Basic's sophisticated debugger in Appendix D allows the instructor flexibility in deciding when to cover this topic.

Solution Manuals. The Student Solutions Manual contains the answer to every odd-numbered exercise. The Instructor Solutions Manual contains the answer to every exercise and programming project. Both solution manuals are in pdf format and can be downloaded from the Publisher's Web site.

Source Code.[①] The programs for all examples and case studies can be downloaded from the Publisher's Web site.

How to Access Instructor and Student Resource Materials

Online Practice and Assessment with MyProgrammingLab™

MyProgrammingLab helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate, personalized feedback, MyProgrammingLab improves the programming competence of beginning students who often struggle with the basic concepts and paradigms of popular high-level programming languages.

A self-study and homework tool, the MyProgrammingLab course for Visual Basic consists of roughly two hundred small practice exercises covering introductory topics such as variables, calculations, decision statements, loops, procedures, arrays, and more. For students, the system automatically detects errors in the logic and syntax of their code submissions and offers targeted hints that enable students to figure out what went wrong—and why. For instructors, a comprehensive gradebook tracks correct and incorrect answers and stores the code inputted by students for review.

For a full demonstration, to see feedback from instructors and students, or to get started using MyProgrammingLab in your course, visit www.myprogramminglab.com.

Instructor Resources[②]

The following protected instructor resource materials are available on the Publisher's Web site at www.pearsonhighered.com/cs-resources. For username and password information, please contact your local Pearson representative.

- Test Item File
- PowerPoint Lecture Slides
- Instructor Solutions Manual
- Programs for all examples, case studies, and answers to exercises and programming projects (Databases, text files, and picture files needed for the exercises are included in the Programs folder.)

① 相关资源可登录华信教育资源网（www.hxedu.com.cn）下载。
② 相关教辅资源的获取方式可参见书末的“教学支持说明”。

Acknowledgments

Many talented instructors and programmers provided helpful comments and constructive suggestions during the many editions of this text and I am most grateful for their contributions. The current edition benefited greatly from the valuable comments of the following reviewers:

Milam Aiken, University of Mississippi
Geoffrey Campbell, Illinois State University
Sherrie Cannoy, North Carolina A&T State University
Joshua Cuneo, Georgia Gwinnett College
Jean Hendrix, University of Arkansas at Monticello
Colin Ikei, Long Beach City College
Ingyu Lee, Troy University
Pati Milligan, Baylor University
Mohammad Rob, University of Houston—Clear Lake
John Robinson, Rutgers University—Camden
Michael Zurad, University of Wisconsin—Parkside

Many people are involved in the successful publication of a book. I wish to thank the dedicated team at Pearson whose support and diligence made this textbook possible, especially Program Manager Carole Snyder and Team Lead Scott Disanno.

I am grateful to Kathy Liszka of the University of Akron for producing the VideoNotes that accompany the book. I would like to thank Kathy Liszka, Anne Bunner, and Howard Lerner for their excellent proofreading. The competence and graciousness of Shylaja Gattupalli at SPi Global made for a pleasant production process.

I extend special thanks to my editor Tracy Johnson. Her ideas and enthusiasm helped immensely with the preparation of the book.

David I. Schneider
dis@math.umd.edu

改编说明

英文原版目录

本书（英文影印改编版）删除了英文原版的 Chapter 1，并将原版 Chapter 8 ~ Chapter 11 的章节顺序进行了调整，具体修改请参见本书目录。

Contents

① 这部分内容（练习题中部分序号为奇数的题目的答案）可在华信教育资源网（www.hxedu.com.cn）下载。

1 Visual Basic, Controls, and Events

1.1 An Introduction to Visual Basic 2015

Visual Basic 2015 is the latest generation of Visual Basic, a language used by many software developers. Visual Basic was designed to make user-friendly programs easier to develop. Prior to the creation of Visual Basic, developing a friendly user interface usually required a programmer to use a language such as C or C++, often requiring hundreds of lines of code just to get a window to appear on the screen. Now the same program can be created in much less time with fewer instructions.

Why Windows and Why Visual Basic?

What people call **graphical user interfaces**, or GUIs, have revolutionized the software industry. Instead of the confusing textual prompts that earlier users once saw, today's users are presented with such devices as icons, buttons, and drop-down lists that respond to mouse clicks. Accompanying the revolution in how programs look was a revolution in how they feel. Consider a program that requests information for a database. Figure 1.1 shows how a program written before the advent of GUIs got its information. The program requests the six pieces of data one at a time, with no opportunity to go back and alter previously entered information. Then the screen clears and the six inputs are again requested one at a time.

```
Enter name (Enter EOD to terminate): Mr. President
Enter Address: 1600 Pennsylvania Avenue
Enter City: Washington
Enter State: DC
Enter Zip code: 20500
Enter Phone Number: 202-456-1414
```

Figure 1.1 Input screen of a pre-Visual Basic program to fill a database.

Figure 1.2 shows how an equivalent Visual Basic program gets its information. The boxes may be filled in any order. When the user clicks on a box with the mouse, the cursor moves to that box. The user can either type in new information or edit the existing information. When satisfied that all the information is correct, the user clicks on the *Write to Database* button. The boxes will clear, and the data for another person can be entered. After all names have been entered, the user clicks on the *Exit* button. In Fig. 1.1, the program is in control; in Fig. 1.2, the user is in control!

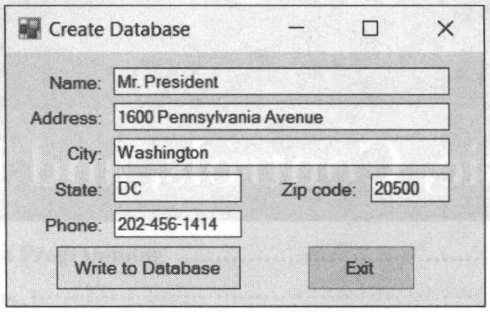


Figure 1.2 Input screen of a Visual Basic program to fill a database.

How You Develop a Visual Basic Program

A key element of planning a Visual Basic program is deciding what the user sees—in other words, designing the user interface. What data will he or she be entering? How large a window should the program use? Where will you place the buttons the user clicks on to activate actions in the program? Will the program have places to enter text (text boxes) and places to display output? What kind of warning boxes (message boxes) should the program use? In Visual Basic, the responsive objects a program designer places on windows are called *controls*. Two features make Visual Basic different from traditional programming tools:

1. You literally draw the user interface, much like using a paint program.
2. Perhaps more important, when you're done drawing the interface, the buttons, text boxes, and other objects that you have placed in a blank window will automatically recognize user actions such as mouse movements and button clicks. That is, the sequence of procedures executed in your program is controlled by "events" that the user initiates rather than by a predetermined sequence of procedures in your program.

In any case, only after you design the interface does anything like traditional programming occur. Objects in Visual Basic recognize events like mouse clicks. How the objects respond to them depends on the instructions you write. You always need to write instructions in order to make controls respond to events. This makes Visual Basic programming fundamentally different from traditional programming. Programs in traditional programming languages ran from the top down. For these programming languages, execution started from the first line and moved with the flow of the program to different parts as needed. A Visual Basic program works differently. Its core is a set of independent groups of instructions that are activated by the events they have been told to recognize. This event-driven methodology is a fundamental shift. The user decides the order in which things happen, not the programmer.

Most of the programming instructions in Visual Basic that tell your program how to respond to events like mouse clicks occur in what Visual Basic calls *event procedures*. Essentially, anything executable in a Visual Basic program either is in an event procedure or is used by an event procedure to help the procedure carry out its job. In fact, to stress that Visual Basic is fundamentally different from traditional programming languages, Microsoft uses the term *project* or *application*, rather than *program*, to refer to the combination of programming instructions and user interface that makes a Visual Basic program possible. Here is a summary of the steps you take to design a Visual Basic program:

1. Design the appearance of the window that the user sees.
2. Determine the events that the controls on the window should respond to.

3. Write the event procedures for those events.

Now here is what happens when the program is running:

1. Visual Basic monitors the controls in the window to detect any event that a control can recognize (mouse movements, clicks, keystrokes, and so on).
2. When Visual Basic detects an event, it examines the program to see if you've written an event procedure for that event.
3. If you have written an event procedure, Visual Basic executes the instructions that make up that event procedure and goes back to Step 1.
4. If you have not written an event procedure, Visual Basic ignores the event and goes back to Step 1.

These steps cycle continuously until the program ends. Usually, an event must happen before Visual Basic will do anything. Event-driven programs are more reactive than active—and that makes them more user friendly.

1.2 Visual Basic Controls

Visual Basic programs display a Windows-style screen (called a **form**) with boxes into which users type (and in which users edit) information and buttons that they click on to initiate actions. The boxes and buttons are referred to as **controls**. In this section, we examine forms and four of the most useful Visual Basic controls.

■ Starting a New Visual Basic Program

Each program is saved (as several files and subfolders) in its own folder. Before writing your first program, you should use File Explorer (with Windows 8 or 10) or Windows Explorer (with Windows 7) to create a folder to hold your programs.

The process for starting Visual Basic varies slightly with the version of Windows and the edition of Visual Studio installed on the computer. Some possible sequences of steps are shown below.

Windows 7 Click the Windows *Start* button, click *All Programs*, and then click on "Microsoft Visual Studio 2015."

Windows 8 and 8.1 Click the tile labeled "Visual Studio 2015." If there is no such tile, click on *Search* in the Charms bar, select the Apps category, type "Visual Studio" into the Search box in the upper-right part of the screen, and click on the rectangle labeled "Visual Studio 2015" that appears on the left side of the screen.

Windows 10 Click the Windows **Start** button, click *All apps*, and then click on *Visual Studio 2015*. Or, click on the tile labeled *Visual Studio 2015*.

Figure 1.3 shows the top part of the screen after Visual Basic is started. A Menu bar and a Toolbar are at the top of the screen. These two bars, with minor variations, are always present while you are working with Visual Basic. The remainder of the screen is called the Start Page. Some tasks can be initiated from the Menu bar, the Toolbar, and the Start Page. We will usually initiate them from the Menu bar or the Toolbar.

The first item on the Menu bar is File. Click on File, hover over (or click on) *New*, and then click on *Project* to produce a New Project dialog box. (Alternately, press Alt/F/N/P or Ctrl+Shift+N.) Figure 1.4 shows a New Project dialog box produced by the Visual Basic Community 2015 edition. Your screen might look somewhat different than Fig. 1.4 even if you are using the Visual Basic Community 2015 edition.

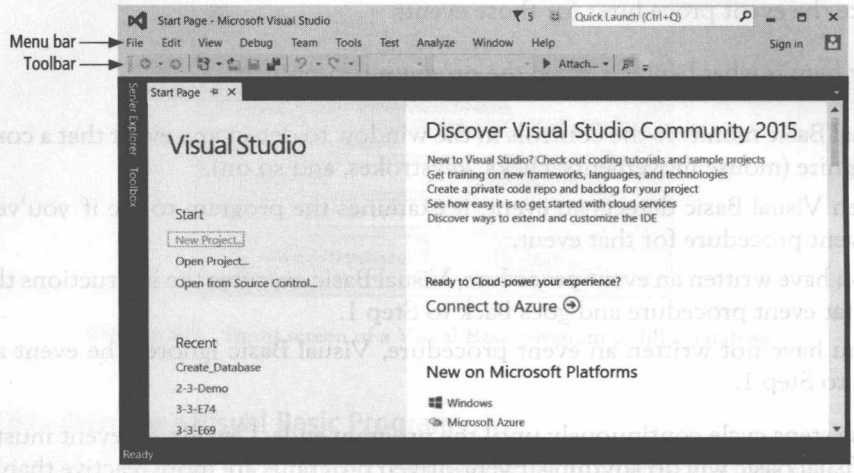


Figure 1.3 Visual Basic opening screen.

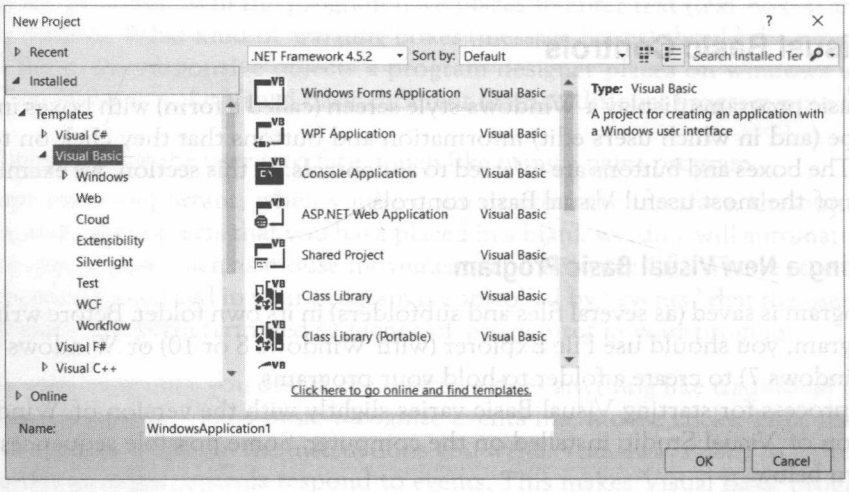


Figure 1.4 The Visual Basic New Project dialog box.

Select *Visual Basic* in the *Templates* list on the left side of Fig. 1.4, and select *Windows Forms Application* in the center list. ***Note:*** The number of items in the center list will vary depending on the edition of Visual Studio you are using.

The name of the program, initially set to WindowsApplication1, can be specified at this time. Since we will have a chance to change it later, let's just use the name WindowsApplication1 for now. Click on the OK button to invoke the Visual Basic programming environment. See Fig. 1.5 on the next page. The Visual Basic programming environment is referred to as the **Integrated Development Environment** or **IDE**. The IDE contains the tools for writing, running, and debugging programs.

It is possible that your screen will look different than Fig. 1.5. The IDE is extremely configurable. Each window in Fig. 1.5 can have its location and size altered. New windows can be displayed in the IDE, and any window can be closed or hidden behind a tab. For instance, in Fig. 1.5 the Toolbox window is hidden behind a tab. The View menu is used to add additional windows to the IDE. If you would like your screen to look similar to Fig. 1.5, click on *Reset Windows Layout* in the Window menu, and then click on the *Yes* button.

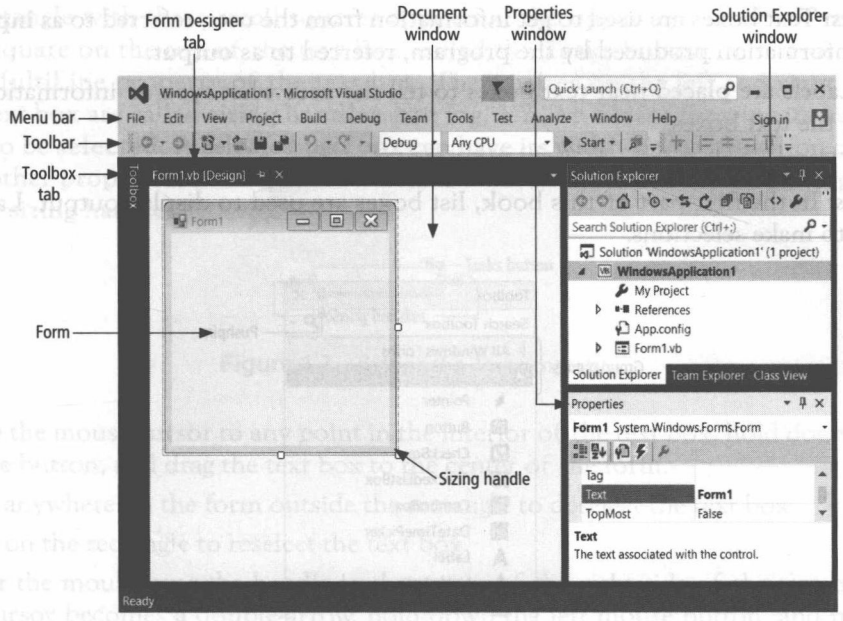


Figure 1.5 The Visual Basic Integrated Development Environment in Form Designer mode.

The **Menu bar** of the IDE displays the menus of commands you use to work with Visual Basic. Some of the menus, like File, Edit, View, and Window, are common to most Windows applications. Others, such as Project, Debug, and Data, provide commands specific to programming in Visual Basic.

The **Toolbar** holds a collection of buttons that carry out standard operations when clicked. For example, you use the sixth button, which looks like two diskettes, to save the files associated with the current program. To reveal the purpose of a Toolbar button, hover the mouse pointer over it. The little information rectangle that pops up is called a **tooltip**.

The **Document window** currently holds the rectangular **Form window**, or **form** for short. (The Form window is also known as the **form designer window** or the **design window**.) The form becomes a Windows window when a program is executed. Most information displayed by the program appears on the form. The information usually is displayed in controls that the programmer has placed on the form. ***Note:*** You can change the size of the form by dragging one of its sizing handles.

The **Properties window** is used to change the initial appearance and behavior of objects on the form. Some (but not all) properties and appearances can be changed by code.

The **Solution Explorer** window displays the files associated with the program and provides access to the commands that pertain to them. (***Note:*** If the Solution Explorer or the Properties window is not visible, click on it in the View menu.)

The **Toolbox** holds icons representing objects (called controls) that can be placed on the form. If your screen does not show the Toolbox, hover the mouse over the Toolbox tab at the left side of the screen. The Toolbox will come into view. Then click on the pushpin icon in the title bar at the top of the Toolbox to keep the Toolbox permanently displayed in the IDE. (***Note:*** If there is no tab marked Toolbox, click on *Toolbox* in the View menu.)

The controls in the Toolbox are grouped into categories such as *All Windows Forms* and *Common Controls*. Figure 1.6 shows the Toolbox after the *Common Controls* group has been expanded. Most of the controls discussed in this text can be found in the list of common controls. (You can obtain a description of a control by hovering the mouse over the control.) The four controls discussed in this chapter are text boxes, labels, buttons, and list boxes. In order to see all the group names, collapse each of the groups.

Text boxes: Text boxes are used to get information from the user, referred to as **input**, or to display information produced by the program, referred to as **output**.

Labels: Labels are placed near text boxes to tell the user what type of information is displayed in the text boxes.

Buttons: The user clicks on a button to initiate an action.

List boxes: In the first part of this book, list boxes are used to display output. Later, they are used to make selections.

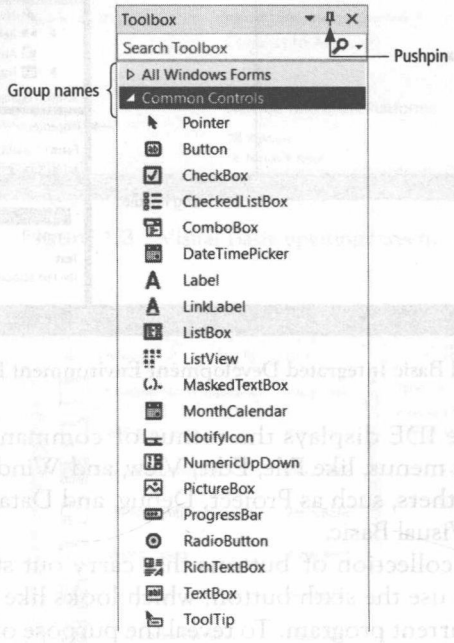


Figure 1.6 The Toolbox's common controls.

An Important Setting

The process of naming and saving programs can proceed in two different ways. In this book, we do not require that a program be given a name until it is saved. The following steps guarantee that Visual Basic will follow that practice.

1. Click on *Options* from the Tools menu to display an Options dialog box.
2. Click on the *Projects and Solutions* item in the left pane of the Options dialog box.
3. If the box labeled "Save new projects when created" is checked, uncheck it.
4. Click on the *OK* button.
5. Open the File menu in the Toolbar and click on *Close Solution*. ***Note:*** If a dialog box appears and asks you if you want to save or discard changes to the current project, click on the *Discard* button.

A Text Box Walkthrough

Place a text box on a form

1. Start a new Visual Basic program.
2. Double-click on the TextBox control (TextBox) in the *Common Controls* group of the Toolbox.

A rectangle with three small squares appears at the upper-left corner of the form. The square on the top of the text box, called the **Tasks button**, can be used to set the MultiLine property of the text box. The squares on the left and right sides of the text box are called **sizing handles**. See Fig. 1.7. An object showing its handles is said to be **selected**. A selected text box can have its width altered, location changed, and other properties modified. You alter the width of the text box by dragging one of its sizing handles.

Figure 1.7 Setting the Text property.

3. Move the mouse cursor to any point in the interior of the text box, hold down the left mouse button, and drag the text box to the center of the form.
4. Click anywhere on the form outside the rectangle to deselect the text box.
5. Click on the rectangle to reselect the text box.
6. Hover the mouse over the handle in the center of the right side of the text box until the cursor becomes a double-arrow, hold down the left mouse button, and move the mouse to the right.

   The text box is stretched to the right. Similarly, grabbing the handle on the left side and moving the mouse to the left stretches the text box to the left. You also can use the handles to make the text box smaller. Steps 2, 3, and 6 allow you to place a text box of any width anywhere on the form. ***Note:*** The text box should now be selected; that is, its sizing handles should be showing. If not, click anywhere inside the text box to select it.
7. Press the Delete key to remove the text box from the form.

   Step 8 gives an alternative way to place a text box of any width at any location on the form.
8. Click on the text box icon in the Toolbox, move the mouse pointer to any place on the form, hold down the left mouse button, drag the mouse on a diagonal, and release the mouse button to create a selected text box.

   You can now alter the width and location as before. ***Note:*** The text box should now be selected. If not, click anywhere inside the text box to select it.

Activate, move, and resize the Properties window

9. Press F4 to activate the Properties window.

   You also can activate the Properties window by clicking on it, clicking on *Properties Window* from the View menu, or right-clicking on the text box with the mouse button and selecting *Properties* from the context menu that appears. See Fig. 1.8. The first line of the Properties window (called the **Object box**) reads "TextBox1", etc. TextBox1 is the current name of the text box. The third button in the row of buttons below the Object box, the *Properties* button ( ), is normally highlighted. If not, click on it. The left column of the Properties window gives the available properties, and the right column gives the current settings of the properties. The first two buttons ( ) in the row of buttons below the Object box permit you to view the list of properties either grouped into categories or alphabetically. You can use the up- and down-arrow keys (or the scroll arrows, scroll box, or the mouse scroll wheel) to move through the list of properties.

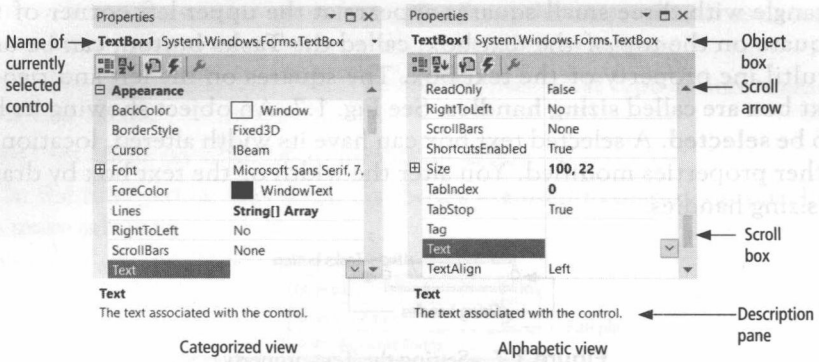


Figure 1.8 Text box Properties window.

10. Click on the Properties window's title bar and drag the window to the center of the screen.

The Properties window is said to be **floating** or **undocked**. Some people find a floating window easier to work with.

11. Drag the lower-right corner of the Properties window to change the size of the Properties window.

An enlarged window will show more properties at once.

12. Hold down the Ctrl key and double-click on the title bar.

The Properties window will return to its original docked location. We now will discuss four properties in this walkthrough.

Set four properties of the text box

Assume that the text box is selected and its Properties window activated.

Note 1: The third and fourth buttons below the Object box, the *Properties* button and the *Events* button, determine whether properties or events are displayed in the Properties window. Normally the *Properties* button is highlighted. If not, click on it.

Note 2: If the Description pane is not visible, right-click on the Properties window, then click on *Description*. The Description pane describes the currently highlighted property.

13. Move to the Text property with the up- and down-arrow keys (alternatively, scroll until the Text property is visible, and click on the property).

The Text property, which determines the words displayed in the text box, is now highlighted. Currently, there is no text displayed in the Text property's Settings box on its right.

14. Type your first name, and then press the Enter key or click on another property. Your name now appears in both the Settings box and the text box. See Fig. 1.9.

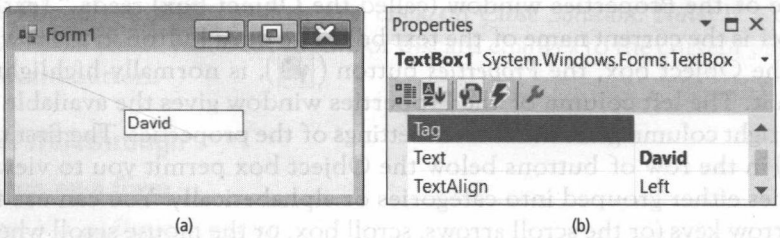


Figure 1.9 Setting the Text property.