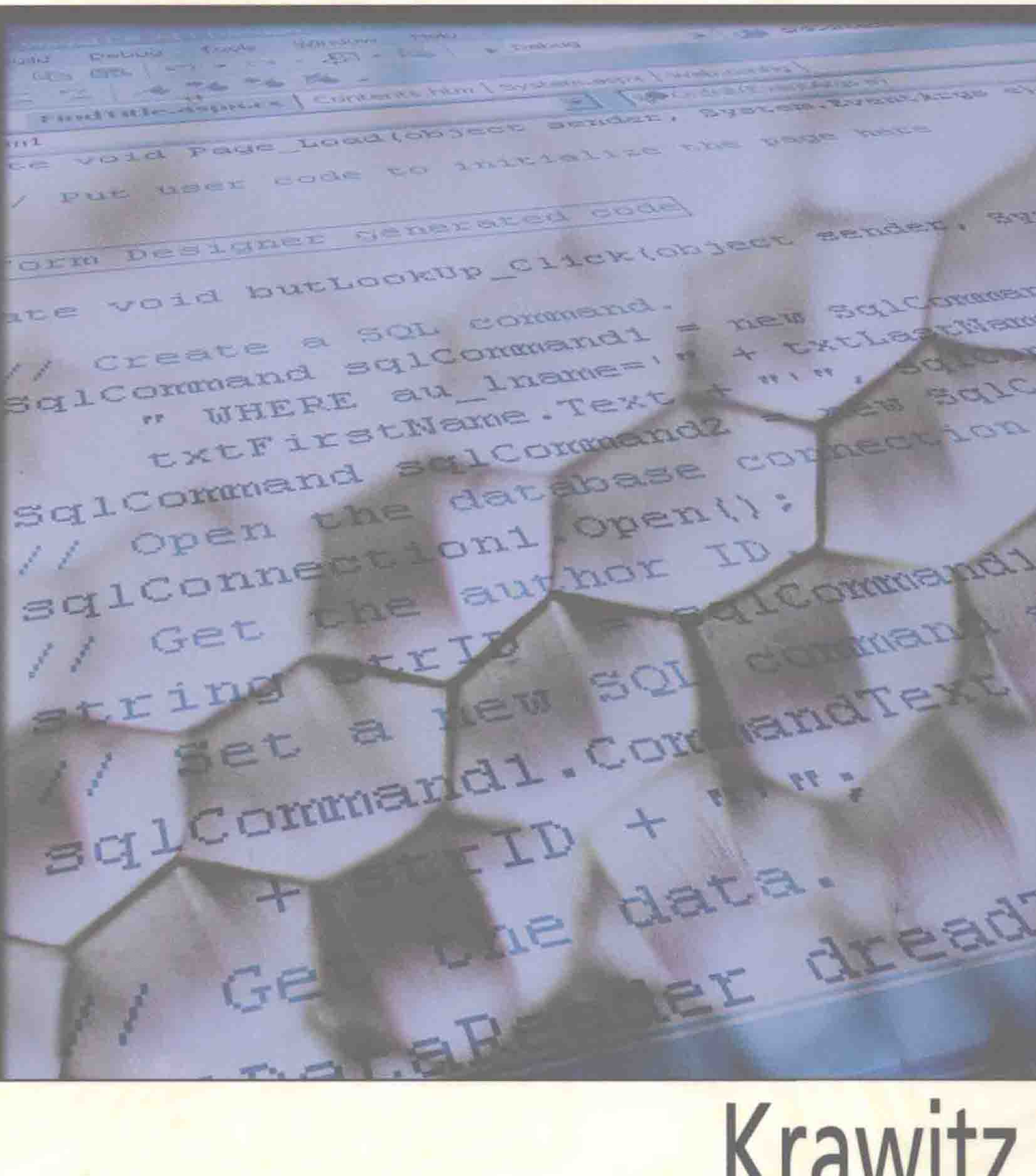
THOMSON
DELMAR LEARNING

Introduction to Programming Using the Tool Visual Basic 2005



CD INSIDE

Krawitz

An Introduction to Programming Using the Tool: Visual Basic 2005

RON KRAWITZ
DeVry University
Phoenix, AZ

THOMSON
DELMAR LEARNING™

Australia Canada Mexico Singapore Spain United Kingdom United States

THOMSON
DELMAR LEARNING

An Introduction to Programming Using the Tool: Visual Basic 2005

Ron Krawitz

Vice President, Technology and Trades SBU
Alar Elken
Editorial Director
Sandy Clark
Senior Acquisitions Editor
Stephen Helba
Development
Ohlinger Publishing Services

Marketing Director
David Garza
Channel Manager
Dennis Williams
Marketing Coordinator
Stacey Wiktorek
Production Director
Mary Ellen Black
Senior Production Manager
Larry Main

Art/Design Coordinator
Francis Hogan
Technology Project Manager
Linda Verde
Senior Editorial Assistant
Dawn Daugherty
Production Services
TIPS Technical Publishing, Inc.

Printed in Canada

2 3 4 5 XXX 08 07 06 05

For more information contact
Delmar Learning
Executive Woods
5 Maxwell Drive, PO Box 8007
Clifton Park, NY 12065-8007

Or find us on the World Wide Web at http://delmar.com

ISBN: 1-4018-7201-8

Dedication:

First and foremost, to my wife, Teri, for her support, encouragement, cajoling and generally making this as easy as possible for me. To my children for their support throughout. To my grandaughter for the inspiration she provides. And finally to my father for giving me the love of learning and teaching that kept me going while I wrote this book.

Preface

PROGRAMMING DEFINED

Programming a computer means telling the computer what to do in a specific sequence. In order to program a computer, you must understand the three programming constructs: Sequence, Decisions, and Loops. Writing a computer program requires using one, two, or all three of the programming constructs to tell the computer what to do. Programmed correctly, the computer will manipulate data and provide valid information that can be used as part of the solution to the problem.

Dictionary.com defines the word program as follows:

1. A listing of the order of events and other pertinent information.
2. An ordered list of events to take place or procedures to be followed.
3. A set of coded instructions that enables a machine, especially a computer, to perform a desired sequence of operations.
4. An instruction sequence in programmed instruction.

Nowhere is a particular programming language mentioned. Once a program is designed, it can be coded into any programming language. The particular language is not important.

PROGRAMMING VS CODING

My students didn't know the difference between designing a program and writing the code to implement the program. They saw programming as task oriented instead of problem solving. The textbooks we used contributed to these misconceptions. I contributed to these misconceptions.

Programming is problem solving. Specifically, problems are solved by creating solutions that run on a computer. In order to create a solution, a programmer must understand the problem, determine what information is available with which to solve the problem, and design a solution that uses some or all of that available information. The solution includes designing and writing a computer program.

A program has three constituent parts: input, process, and output. Input and output are important to make the program useful but most of the work gets done in process. Process is the heart of programming. Process does not include user interface; therefore, process does not include GUI.

PROGRAMMING LANGUAGES

A few years ago Microsoft introduced a new programming language, C# (C sharp). C# is part of Microsoft's .NET (dot net) family of programming languages. It took me about one hour to become reasonably proficient in C#.NET. Why? Because I know how to program. The programming language is merely a new syntax.

While writing this book about Visual Basic, I was also teaching C++ (C plus plus), C#, and Java. I could switch back and forth among the programming languages readily and easily. Did I sometimes use the syntax of one language while coding in another? Yes! When that happened, I told my class, "Sorry, that's C#," and changed the code. I develop solutions to problems using available resources without being restricted to one or two programming languages.

Practice with a programming language improves efficiency. However, anywhere from a few hours to a few weeks with a new programming language is all that is necessary to become proficient. Programming languages come and go, but program developers will be around for quite a while.

ABOUT THE BOOK

The primary purpose of this book is to teach programming techniques. However, it does so in a new way. Instead of focusing on a single programming language, it focuses on generic programming techniques. Programming techniques are pretty much universal. The book teaches Visual Basic 2005, but where this language differs from other programming languages is carefully explained. Once you learn the basics of programming in one language, you can usually program in another language with very little additional learning.

When I started teaching, I followed the proscribed textbook chapter by chapter. Students could successfully complete a task (write a loop that sequences from -11 to -236 in steps of 11 and display the loop variable each time the loop executes), but they could not create a program that included a loop.

The text had made them "experts" in a programming language. They could create programming constructs in that language as long as they had sufficient instructions. However, they did not know how to solve a problem by developing a program. The course was "Visual Basic" to them, not "Introduction to Programming." The particular syntax was more important than the concept.

My students also confused graphical user interface (GUI) with developing solutions. The visual display a program presents is very important. However, this visual display is secondary to developing the program. Students could create and manipulate GUI well, but they confused GUI with programming.

An Introduction to Programming Using the Tool: Visual Basic 2005 focuses on what a loop is, what it does, and when it is used. It demonstrates where to use a loop, shows how to implement a loop in Visual Basic 2005, provides practice in writing loops in Visual Basic 2005, and shows that Visual Basic 2005 is just one of many tools with which a loop may be implemented.

Every introduction to programming book that utilizes Visual Basic 2005 begins with GUI and the many things that go into manipulating it. Although it is useful and visually pleasing, GUI isn't the basis of programming.

Teaching GUI takes time. To make the time it takes to teach and practice GUI as productive as possible, I chose to teach the basic programming concepts using Console Applications (no GUI) instead of Windows Applications (GUI). In this way, more time is spent learning how to program rather than learning about GUI.

GUI is introduced in the second half of the book so that students can continue practicing the basics of programming while implementing GUI. In this way, as students learn the various visual objects, they can create programs behind the GUI to implement more complex solutions—practicing programming while learning GUI.

Teaching programming without emphasizing a specific programming language became my goal. I wanted to demonstrate generic programming without focusing on a particular language. I wanted to emphasize what is generic and what is particular to an individual programming language.

Students crave the feedback they get from actually running a program. Just look at their faces the first time they write a program that runs. In order to run a program they need a programming language. I chose a programming language that is easy to use. An Introduction to Programming Using The Tool: Visual Basic 2005 demonstrates and teaches programming techniques with an emphasis on the generic while working with a specific tool, VB 2005.

This book presents the basics of programming early in the text and adds GUI, design, and other concepts later. In this way, students can practice programming while learning GUI. This is in stark contrast to practicing GUI while learning programming.

I hold off program design until after the basic programming constructs are established. This allows students to learn how to implement solutions to simple problems that actually require some thought and design. In my years of teaching I have seen far too many students who believe design is supposed to be done immediately after the code has been debugged and tested.

Designing a solution to a problem with a computer program is no different than acquiring any other skill; it must be practiced and practiced. When learning how to use the various programming constructs, students must read about the technique, see the technique, and then practice the technique. This book explains the techniques, demonstrates the techniques, and provides an opportunity to practice the techniques.

The term developer should (and may eventually) replace programmer. The word programmer is too often confused with coder. Programmers develop solutions. They may or may not actually write the code. Using this book will help students understand that their value is the creativity they bring to developing the solution. Knowledge of coding will increase their worth as problem solvers. They need to see themselves as problem solvers, not coders.

Most students will move from this course to a course in Object Oriented Programming (OOP). Everything in this book sets the stage for OOP. The basics of OOP are laid out without actually being used. Students leave this course ready to learn OOP.

While writing, I gave working copies of this book to students to use in their courses. They really liked it. One student, who had to drop programming due to illness, used this book from beginning to end and successfully passed the course. She began using both the assigned text and this book, but stopped using the assigned text within two weeks. I hope you have the same success.

SUPPLEMENTAL MATERIALS

Many of the programs used in this book are provided to students in an included supplemental CD. Please note that the programs cannot be run directly from this CD. They must be copied to a writable part of your computer in order to execute. You will then be able to make changes to the programs by "playing" with the programming language. Feel free to change whatever you wish. When you are finished, you can always restore the program to its original form by copying from the CD.

The Instructor's E.Resource CD contains:

- a computerized test bank that includes 30 multiple choice questions per chapter
- answers to the book's Challenge Problems
- an Image Library with many of the book's figures, which can be transferred into PowerPoint for classroom presentations

ABOUT THE AUTHOR

Ron is currently a professor of Computer Information Systems (CIS) at DeVry University's Phoenix Campus. In addition to teaching CIS he also teaches courses in the Computer Engineering Technology program.

His background marries the pragmatic, profit oriented, industrial world with the more theoretical, knowledge based, academic world. He is an Electrical Engineer who began his career by designing and programming specialty computers. Along the way he learned several programming languages while beginning to understand how and why computers can benefit people in both the industrial and educational environment.

Ron received his Bachelor of Science degree from the University of Michigan and his Masters of Business Administration from Widener University. Two of the highlights of his career occurred in 1999 when two previous clients contacted him to find out if programs written in the 1970s were Y2K compliant. He never dreamed any of his programs would continue to be used for 30 years.

Ron began teaching in the classroom in 1988. He has taught business, CIS, computer science, physics, math, and engineering courses over the years. He is also a business consultant who keeps current with information systems and the computer science world by spending time among a variety of professionals. Ron was a professional soccer coach for 10 years and spends his present off-hours directing and stage managing plays at local theaters.

ACKNOWLEDGMENTS

Current and previous students have given me tremendous support in writing this book. Romony Heng and Dan Flumerfelt, in particular, helped me create the chapter questions and verified answers throughout the book.

I would also like to thank the following reviewers whose comments helped in the development of my book:

Harold Broberg, Indiana Purdue University, Fort Wayne, IN

David Courtaway, DeVry University, Pomona, CA

Junior Joesph Gentles, DeVry University, Pomona, CA

Keith Risman, Virginia College, Birmingham, AL

Frank Lanzer, Anne Arundel Community College, Arnold, MD

Table of Contents

Preface xiii

Part I Programming 1

Chapter 1 Introduction to Programming 3

1.1: A Programmer's Goals 4
1.2: Data and Information 4
1.3: Introduction to Programming 5
1.4: Programming Constructs 6
1.5: Design Time and Run Time 7
1.6: Programming Errors 7
1.7: Variables 8
1.8: Arithmetic Expressions 12
1.9: Relational Expressions 13
1.10: The Boolean Data Type 14
1.11: Logical Expressions 14

Chapter 2 Programming Sequence 25

2.1: Computers Are Intelligent—or Are They? 25
2.2: One at a Time 26
2.3: Keywords 27
2.4: Sub-Procedures and Functions 28
2.5: The Assignment Operator 33
2.6: Equations 33
2.7: Counting 35

Part II Visual Basic 2005 Console Applications 43

Chapter 3 Starting Visual Basic 2005 for a Console Application 45

3.1: Starting Visual Studio 2005 46
3.2: Creating a Visual Basic 2005 Project 47

3.3: Understanding the Parts of the Visual Basic 2005 Console Application IDE 49
3.4: Good Programming Practices and Procedures 52
3.5: Outputting Information from a VB 2005 Program 63
3.6: Console.Out.WriteLine( ) 64
3.7: Console.Out.Write( ) 66
3.8: Customizing Console.Out.Write( ) and Console.Out.WriteLine( ) Arguments 68
3.9: Inputting Data to a VB 2005 Program 71
3.10: The Little Old Lady 74
3.11: Back to Inputting Data to a Program 76
3.12: Parsing 77

Part III Introduction to Programming Continued . . .87

Chapter 4 Decision Making 89
4.1: Decisions 90
4.2: If Then 91
4.3: If Then Else 97
4.4: Select Case 106
4.5: Comparing Strings 109
4.6: Nested Statements 117
4.7: Comparing Numbers that Contain Decimals 120

Chapter 5 Looping 129
5.1: Infinite Loops 130
5.2: Finite Loops 133
5.3: While Loop 135
5.4: For Loop 147
5.5: Nested Loops 157
5.6: Elementary File Operations 157
5.7: Random Numbers 164

Chapter 6 Arrays 173
6.1: Why Arrays? 175
6.2: Declaring Arrays in VB Using the Dimension Statement 176
6.3: Identiying the Individual Parts of an Array 177
6.4: How Computers Store Variables 182
6.5: Special Case—Arrays in Visual Basic 2005 183

6.6: Array Functions 184
6.7: Initializing Arrays 187
6.8: One-Dimensional Arrays 189
6.9: Relating Multiple Pieces of Data to Each Other 193
6.10: Two-Dimensional Arrays 194
6.11: Arrays of Three or More Dimensions 196
6.12: Getting Things in the Right Order 198
6.13: Test Driving What You Have Learned 201

Chapter 7 Designing Console Solutions 213
7.1: "The Rest Is Just Syntax" 214
7.2: A Computer Program 217
7.3: Program Design 219
7.4: Designing a Solution 226
7.5: Stubs 229
7.6: Modeling Diagrams 229
7.7: Flowcharts 230
7.8: Designing the Solution 246
7.9: Test Driving What You Have Learned 247

Part IV Introduction to GUI Programming 261

Chapter 8 Visual Basic 2005 GUI Programming 263
8.1: Starting Visual Basic 2005 for a Windows Application 264
8.2: GUI Programming 265
8.3: What is Object Oriented Programming (OOP)? 266
8.4: Class Diagrams and Abstract Data Types 267
8.5: Reusable Code 268
8.6: Your VB 2005 Program 271
8.7: Toolbox 273
8.8: VB 2005 Toolbox Items—Classes 273
8.9: Properties 274
8.10: Programs with Many Subs 286
8.11: Lifetime and Scope 286
8.12: Events 288
8.13: Calling Functions or Subs 290
8.14: A Listing of All the Objects on Your Form 292
8.15: Forms and Classes 293

8.16: Startup Form 300

8.17: Visual Basic Notation 305

Chapter 9 Designing GUI Solutions 317

9.1: Flowcharting a GUI Solution 318

9.2: The GUI Program Design–the Detail 323

9.3: The GUI Program Design–the Concept 338

9.4: Programming Efficiency 338

9.5: Input Validation 344

Chapter 10 True – False Test Administrator 365

10.1: Solving the Problem—Step by Step 366

10.2: Adding a Timer 384

10.3: Only the First Answer Will Be Accepted 387

10.4: What if the User Changes His or Her Mind? 390

Chapter 11 True – False Test II 399

11.1: Using the True – False Test Program for a C# Test 400

11.2: More than One Form in a Program 401

11.3: Class Declaration 403

11.4: The Second Form—frmTestAdministrator from Listing 10.3 404

11.5: "set" Subs and "get" Functions 406

11.6: Back to True – False Test 408

11.7: Loading the Tests into a File 411

11.8: Finishing the True – False Test Administrator Program 418

Chapter 12 Searching and Ordering a Data Collection 425

12.1: Creating a Data Collection 426

12.2: Searching a Data Collection 446

12.3: Ordering Data Collections 464

12.4: Finding an Element in an Ordered Collection 468

12.5: GoFish v1 478

12.6: A Word of Caution 484

Chapter 13 Inserting into and Deleting from a Data Collection 489

13.1: Inserting a Record into a Data Collection 491

13.2: Deleting a Record from a Data Collection 509

13.3: Putting the Pieces Together to Create a Program 512

13.4: The Program 518

Chapter 14 Debugging Programs .527

14.1: Syntax Errors 528

14.2: Logic Errors 532

14.3: The Debugger 551

14.4: General Debugging Techniques 555

Chapter 15 Object Oriented Programming 567

15.1: Why OOP? 569

15.2: The Three Principles of OOP 571

15.3: OOP Concepts 578

15.4: Putting It All Together 594

Glossary G-1

Index I-1

PART I

Programming

PART OUTLINE

Chapter 1 Introduction to Programming
Chapter 2 Programming Sequence

Part I introduces basic programming techniques, with an emphasis on generic techniques rather than a specific programming language. It explores the specific areas needed to develop a program. Part I also demonstrates sequence, one of the three programming constructs.

In order to provide information needed to solve a problem a computer program must provide a sequential series of instructions. This series of instructions is called a program. Before you can learn all of the intricacies of providing the information needed to solve a problem you have to learn how to provide the instructions the computer needs to complete its task. Part I begins this process.

CHAPTER 1

Introduction to Programming

OBJECTIVES

- Explain What a Program Is and Why It Is Written
- Define the Three Programming Constructs: Sequence, Decisions, and Looping
- Describe the Difference between Design Time and Run Time
- Identify the Two Different Types of Programming Errors
- Explain Arithmetic, Relational, and Logical Expressions

INTRODUCTION

A computer is a device that computes, especially a programmable electronic machine that performs high-speed mathematical or logical operations or that otherwise processes information. How does a computer do this? Modern computers are full of electronic devices that, if instructed properly, can transform data into useful information.

The key words in this definition are "if instructed properly". Without proper instructions the computer is little more than a good paper weight. The computer itself is analogous to a body with no brain; it can sit there and look good but it can't do anything on its own. The "brain", the thing that makes the computer work, is the computer program.

A computer program is a sequence of instructions that a computer can interpret and execute. The trick is in providing the right instructions in the correct order to accomplish a specific task. This book will teach you how to create and where to put those instructions.

Businesses do not want computer programs—businesses want what the computer can do once it is programmed properly. Most businesses have significantly more invested in computer programs, or software, than they have invested in the computer, or hardware. The health and longevity of their software is very important to their profitability.

A computer takes data from files, which are stored inside the computer, and inputs, from a keyboard, mouse, or other device, and manipulates it so that data becomes meaningful. The outputs—the information shown on the screen or printed from a printer—can then be used to do something beneficial for the business.

For example, a clerk working in a grocery store is trying to decide how much to charge for a can of tuna. The clerk scans the barcode on the can (input). The computer program tells the computer to look up the price (from a file), computes the cost of the tuna, and displays the cost on a screen (output). Because the computer was properly programmed, or directed, it was able to generate information that was helpful to the business.

CHAPTER OUTLINE

1.1 A Programmer's Goals
1.2 Data and Information
1.3 Introduction to Programming
1.4 Programming Constructs
1.5 Design Time and Run Time
1.6 Programming Errors
1.7 Variables
1.8 Arithmetic Expressions
1.9 Relational Expressions
1.10 The Boolean Data Type
1.11 Logical Expressions

1.1 A PROGRAMMER'S GOALS

Whenever a programmer is called upon to solve a problem with a computer program s/he has two equally important goals.

Goal #1: The program must provide accurate and correct information that can be used to solve the problem. No matter what else it does, the program must provide information to solve the basic problem for which it was created. If this basic goal is not met, the program is a waste of time.

Goal #2: The program must be maintainable or modifiable by another programmer. Eventually, even the most well written program will have to be modified so it can adapt to changing needs and continue working properly. If this basic goal is not met, the program is a waste of time.

Everything in this book will adhere to both of these goals.

QUICK REVIEW

1. T F One of a programmer's goals is to provide accurate and correct information that can be used to solve a given problem.
2. T F One of a programmer's goals is to get the correct answer, regardless of the quality of the input information.
3. T F One of a programmer's goals is to be able to "read" another programmer's program so it can be used as intended.
4. T F One of a programmer's goals is to create a maintainable and modifiable program.
5. T F Over time a program may fail to work as intended.
6. T F A well written program is designed to work no matter what changes happen to the computer or programming language.
7. T F Programmers show respect for each other by refusing to work on another programmer's program.
8. T F If the program can be written in less time, it is acceptable to disregard the programmer's goals.

1.2 DATA AND INFORMATION

Data is what is put into a computer program in an "undigested," or raw form. Data includes numbers, letters, names, or words. As the computer program "digests" the data it sorts, manipulates, culls, and calculates the data into something organized and meaningful. Once the data is organized it becomes information. **Information** is what you get from a properly written program.

QUICK REVIEW

9. T F Data and information are interchangeable.
10. T F Data is what a computer produces when it has completed its task.