
Preface

Live in fragments no longer. Only connect.

Edward Morgan Forster

We wove a web in childhood,

A web of sunny air.

Charlotte Brontë

Welcome to Visual C++ .NET and the world of Windows, Internet and World-Wide-Web programming with Visual Studio® .NET and the .NET platform! At Deitel & Associates, we write college-level programming-language textbooks and professional books. Writing *Visual C++ .NET How to Program* was a joy. This book and its support materials have everything instructors and students need for an informative, interesting, challenging and entertaining educational experience. Also, we have included a tour of the book in this preface to help instructors, students and professionals get a sense of the rich coverage this book provides of Visual C++ .NET programming.

In this preface, we overview the conventions we use in *Visual C++ .NET How to Program*, such as syntax coloring the code examples, “code washing” and highlighting important code segments to help focus students’ attention on key concepts introduced in each chapter. We also overview the features of *Visual C++ .NET How to Program*.

This textbook is up-to-date with Microsoft’s latest release of Visual Studio—Visual Studio .NET 2003, which includes an updated version of Visual C++ .NET. Every example and exercise solution was built and tested using the Visual C++ .NET Standard Edition version 2003 software. For the educational market only, this textbook is available in a “value pack” with the Microsoft® Visual C++® .NET Standard Edition version 2003 integrated development environment as a free supplement. The standard edition is fully functional and is shipped on 5 CDs. There is no time limit for using the software. [*Note: Professionals using this publication will have to purchase the necessary software to build and run the applications in this textbook.*]

We discuss *Visual C++ .NET How to Program*'s comprehensive suite of educational materials that help instructors maximize their students' learning experience. These include an Instructor's Resource CD with solutions to the book's chapter exercises and a Test-Item File with hundreds of multiple-choice examination questions and answers. Additional instructor resources are available at the book's Companion Web Site (www.prenhall.com/deitel), which includes a Syllabus Manager and customizable PowerPoint® Lecture Notes. PowerPoint slides and additional support materials are available for students at the Companion Web Site, as well.

Visual C++ .NET How to Program was reviewed by a team of distinguished academics and industry professionals. We list their names and affiliations so you can get a sense of how carefully this book was scrutinized. The preface concludes with information about the authors and about Deitel & Associates, Inc. As you read this book, if you have any questions, please send an e-mail to deitel@deitel.com; we will respond promptly. Please visit our Web site, www.deitel.com, regularly and be sure to sign up for the *DEITEL® Buzz Online* e-mail newsletter at www.deitel.com/newsletter/subscribe.html, which provides information about our publications, company announcements, links to informative technical articles, programming tips, teaching tips, challenges and anecdotes.

Features of Visual C++ .NET How to Program

This book contains many features, including:

Syntax Highlighting

We syntax highlight all the Visual C++ .NET code in a manner similar to that of Visual Studio .NET. This greatly improves code readability—an especially important goal, given that this book contains over 22,000 lines of code. Our syntax-highlighting conventions are as follows:

```
comments appear like this
keywords appear like this
constants and literal values appear like this
errors and ASP .NET directives appear like this
all other code appears in black
```

Code Highlighting and User-Input Highlighting

We have added extensive code highlighting to make it easier for readers to spot the featured segments of each program. This feature also helps students review the material rapidly when preparing for exams or labs. We have also highlighted in our screen dialogs all user inputs to distinguish them from program outputs.

“Code Washing”

This is our term for applying comments, using meaningful identifiers, applying indentation and using vertical spacing to separate meaningful program units. This process results in programs that are much more readable and self-documenting. We have added descriptive comments to all of the code to help the student clearly understand the flow of the program. We have done extensive code washing of all the source code programs in the text and the ancillaries.

Web Services

Microsoft's .NET strategy embraces the Internet and Web as integral to software development and deployment. Web-services technology enables information sharing, e-commerce and other interactions using standard Internet protocols and technologies, such as Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML) and Simple Object Access Protocol (SOAP). Web services enable programmers to package application functionality in a manner that turns the Web into a library of reusable software components. Chapter 20, *Web Services*, presents a Web service that allows users to manipulate “huge integers”—integers too large to be stored in .NET's built-in data types. In this example, a user enters two huge integers and presses buttons to invoke a Web service that adds, subtracts or compares the two huge integers.

Object-Oriented Programming

Object-oriented programming is the most widely employed technique for developing robust, reusable software. This text offers a rich treatment of Visual C++ .NET's object-oriented programming features. Chapter 8, Object-Based Programming, introduces how to create classes and objects. These concepts are extended in Chapter 9, Object-Oriented Programming: Inheritance, which discusses how programmers can create powerful new classes quickly by “absorbing” the capabilities of existing classes. Chapter 10, Object-Oriented Programming: Polymorphism, focuses on the relationships between classes in a hierarchy.

XML

Extensible Markup Language (XML) use is exploding in the software-development industry, in the e-business and e-commerce communities, and is pervasive throughout the .NET platform. Because XML is a platform-independent technology for describing data and for creating markup languages, XML's data portability integrates well with Visual C++ .NET based applications and services. Chapter 18, Extensible Markup Language (XML), introduces XML. In this chapter, we present XML markup and discuss the Document Object Model (DOM[™]), which is used to manipulate XML documents programmatically.

Multithreading

Computers enable programmers to perform many tasks in parallel (i.e., concurrently), such as printing documents, downloading files from a network and surfing the Web. Multithreading is the technology through which programmers can develop applications that perform concurrent tasks. Historically, a computer has contained a single, expensive processor, which its operating system would share among all applications. Today, processors are becoming increasingly inexpensive, making it possible to build affordable computers with many processors working in parallel—such computers are called multiprocessors. Multithreading is effective on both single-processor and multiprocessor systems. .NET's multithreading capabilities make the platform and its related technologies better prepared to handle today's sophisticated multimedia-intensive, database-intensive, network-based, multiprocessor-based, distributed applications. Chapter 14, Multithreading, introduces this powerful capability.

ADO .NET

Databases store vast amounts of information that individuals and organizations must access to conduct business. As an evolution of Microsoft's ActiveX Data Objects (ADO) technology, ADO .NET represents a new approach to building applications that interact with da-

tabases. ADO .NET uses XML and an enhanced object model to provide developers with the tools they need to access and manipulate databases for large-scale, extensible, mission-critical, multi-tier applications. Chapter 19, Database, SQL and ADO .NET, introduces ADO .NET and the Structured Query Language (SQL) for manipulating databases.

Graphical User Interfaces

Visual Studio .NET 2003 and Visual C++ .NET Standard Edition version 2003 include a *Windows Form designer*. This form designer simplifies the creation of graphical user interfaces (GUIs) for the programmer by generating the GUI code. GUI applications are programs that display graphical elements, such as buttons and labels, with which the user interacts. This book presents many GUI applications to demonstrate different topics in Visual C++ .NET. Visual Studio .NET contains built-in GUI development tools for C#, Visual Basic[®] .NET and Visual C++ .NET.

XHTML™

The World Wide Web Consortium (W3C) has declared HTML to be a legacy technology that will undergo no further development. HTML is being replaced by the Extensible Hypertext Markup Language (XHTML)—an XML-based technology that is rapidly becoming the standard for describing Web content. We use XHTML in Chapter 18, Extensible Markup Language (XML), and offer an introduction to the technology in Appendix E, Introduction to XHTML: Part 1 and Appendix F, Introduction to XHTML: Part 2. These appendices overview headers, images, lists, image maps and other XHTML features.

Unicode[®]

As computer systems evolved worldwide, computer vendors developed numeric representations of character sets and special symbols for the local languages spoken in different countries. In some cases, different representations were developed for the same languages. Such disparate character sets hindered communication among computer systems. Visual C++ .NET supports the *Unicode Standard* (maintained by a non-profit organization called the *Unicode Consortium*), which maintains a single character set that specifies unique numeric values for characters and special symbols in most of the world's languages. Appendix D, Unicode, discusses the standard, overviews the Unicode Consortium Web site, www.unicode.org and presents a Visual C++ .NET application that displays “Welcome to Unicode!” in several languages.

Bit Manipulation

Computers work with data in the form of binary digits, or bits, which can assume the values 1 or 0. Computer circuitry performs various simple bit manipulations, such as examining the value of a bit, setting the value of a bit and reversing a bit (from 1 to 0 or from 0 to 1). Operating systems, test-equipment, networking software and many other kinds of software require that programs communicate “directly with the hardware” by using bit manipulation. Appendix I, Bit Manipulation, overviews the bit manipulation capabilities that the .NET Framework provides.

Teaching Approach

Visual C++ .NET How to Program contains a rich collection of examples, exercises and projects drawn from many fields and designed to provide students with a chance to solve

interesting, real-world problems. The book concentrates on the principles of software engineering and stresses program clarity. We avoid arcane terminology and syntax specifications in favor of teaching by example. Our code examples have been fully tested using Microsoft's Visual C++ .NET Standard Edition version 2003 software.

We are educators who teach edge-of-the-practice topics in industry classrooms worldwide. This text emphasizes good pedagogy.

Font Conventions

We use fonts to distinguish between IDE features (such as menu names and menu items) and other elements that appear in the IDE. Our convention is to emphasize IDE features in a sans-serif bold **Helvetica** font (for example, **Properties** window) and to emphasize program text in a serif *Lucida* font (`int x = 5`).

Learning Visual C++ .NET with the Live-Code Approach

Visual C++ .NET How to Program is loaded with live-code examples (provided on the accompanying CD)—each new concept is presented in the context of a complete, working program that is immediately followed by one or more sample executions showing the program's input/output dialog. This style exemplifies the way we teach and write about programming. We call this method of teaching and writing the LIVE-CODE approach. *We use programming languages to teach programming languages.* Reading the examples in the text is much like typing and running them on a computer.

World Wide Web Access

All of the source-code examples for *Visual C++ .NET How to Program* (and our other publications) are available on the Internet as downloads from the following Web sites:

www.deitel.com
www.prenhall.com/deitel

Registration is quick and easy and the downloads are free. We suggest downloading all the examples, then running each program as you read the corresponding text. Making changes to the examples and immediately seeing the effects of those changes is a great way to enhance your learning experience. Any instructions we provide for running this book's examples assume that the user is running Windows 2000 or Windows XP and is using Microsoft's Internet Information Services (IIS). Additional setup instructions for IIS and other software can be found at our Web sites—www.deitel.com and www.prenhall.com/deitel—along with the examples. [Note: This is copyrighted material. Feel free to use it as you study, but you may not republish any portion of it in any form without explicit permission from Prentice Hall and the authors.]

Objectives

Each chapter begins with a statement of objectives. This tells the student what to expect and gives the student an opportunity, after reading the chapter, to determine if he or she has met these objectives. It is a confidence builder and a source of positive reinforcement.

Quotations

The learning objectives are followed by a series of quotations. Some are humorous, some are philosophical and some offer interesting insights. Our students enjoy relating the quo-

tations to the chapter material. You may appreciate some of the quotations more *after* reading the chapters.

Outline

The chapter outline enables students to approach the material in top-down fashion. This, too, helps students anticipate future topics and set a comfortable and effective learning pace.

22,000+ Lines of Code in 187 Example Programs (with Program Outputs)

We present Visual C++ .NET features in the context of complete, working programs. The programs range in size from just a few lines of code to substantial examples containing hundreds of lines of code. All examples are available on the accompanying CD and as downloads from our Web site, www.deitel.com.

884 Illustrations/Figures

An abundance of charts, line drawings and program outputs is included.

417 Programming Tips

We have included programming tips to help readers focus on important aspects of program development. We highlight hundreds of these tips in the form of *Good Programming Practices*, *Common Programming Errors*, *Error-Prevention Tips*, *Performance Tips*, *Portability Tips*, *Software Engineering Observations* and *Look-and-Feel Observations*. These tips and practices represent the best the authors have gleaned from many decades of programming and teaching experience. One of our customers—a mathematics major—told us that she feels this approach is like the highlighting of axioms, theorems and corollaries in mathematics books; it provides a foundation on which to build good software.



72 Good Programming Practices

Good Programming Practices *are tips for writing clear programs. These techniques help students produce programs that are more readable, self-documenting and easier to maintain.*



135 Common Programming Errors

Students learning a language—especially in their first programming course—tend to make certain kinds of errors frequently. Focusing on these Common Programming Errors helps students avoid making the same errors. It also helps reduce long lines outside instructors' offices during office hours!



31 Error-Prevention Tips

When we first designed this “tip type,” we thought we would use it strictly to tell people how to test and debug programs. In fact, many of the tips describe aspects of Visual C++ .NET that reduce the likelihood of “bugs” and thus simplify the testing and debugging processes.



49 Performance Tips

In our experience, teaching students to write clear and understandable programs is by far the most important goal for a first programming course. But students want to write the programs that run the fastest, use the least memory, require the smallest number of keystrokes, or dazzle in other nifty ways. Students really care about performance. They want to know what they can do to “turbo charge” their programs. So we highlight opportunities for improving program performance—making programs run faster or minimizing the amount of memory that they occupy.



10 Portability Tips

Software development is a complex and expensive activity. Organizations that develop software must often produce versions customized to a variety of computers and operating systems. So there is a strong emphasis today on portability, i.e., on producing software that will run on a variety of computer systems with few, if any, changes.



104 Software Engineering Observations

The object-oriented programming paradigm necessitates a complete rethinking of the way we build software systems. Visual C++ .NET is effective for achieving good software engineering. The Software Engineering Observations highlight architectural and design issues that affect the construction of software systems, especially large-scale systems.



16 Look-and-Feel Observations

We provide Look-and-Feel Observations to highlight graphical-user-interface conventions. These observations help developers design attractive, user-friendly graphical user interfaces that conform to industry norms.

Summary (759 Summary bullets)

Each chapter ends with additional pedagogical devices. We present an extensive, bullet-list-style *Summary* in every chapter. This helps the student review and reinforce key concepts. There is an average of 35 summary bullets per chapter.

Terminology (2225 Terms)

We include a *Terminology* section with an alphabetized list of the important terms defined in the chapter—again, further reinforcement. There is an average of 101 terms per chapter. Each term also appears in the index, so the student can locate terms and definitions quickly.

507 Self-Review Exercises and Answers (Count Includes Separate Parts)

Extensive *Self-Review Exercises* and *Answers to Self-Review Exercises* are included for self study. This gives the student a chance to build confidence with the material and prepare to attempt the regular exercises.

174 Exercises (Solutions in Instructor's Manual; Count Includes Separate Parts)

Each chapter concludes with a substantial set of exercises including simple recall of important terminology and concepts; writing individual program statements; writing small portions of functions and Visual C++ .NET classes; writing complete functions, Visual C++ .NET classes and programs; and writing major term projects. The large number of exercises enables instructors to tailor their courses to the unique needs of their audiences and to vary course assignments each semester. Instructors can use these exercises to form homework assignments, short quizzes and major examinations. **[NOTE: Please do not write to us requesting the instructor's manual. Distribution of this publication is strictly limited to college professors teaching from the book. Instructors may obtain the solutions manual from their regular Prentice Hall representatives. We regret that we cannot provide the solutions to professionals.]**

Approximately 5,838 Index Entries (with approximately 7,261 Page References)

We have included an extensive *Index* at the back of the book. This helps the student find any term or concept by keyword. The *Index* is useful to people reading the book for the first

time and is especially useful to practicing programmers who use the book as a reference. Most of the terms in the *Terminology* sections appear in the *Index* (along with many more index entries from each chapter). Thus, the student can use the *Index* in conjunction with the *Terminology* sections to be sure he or she has covered the key material of each chapter.

Software Included with Visual C++ .NET How to Program

For the educational market only, this textbook is available in a “value pack” with the Microsoft Visual C++ .NET Standard Edition version 2003 integrated development environment as a free supplement. The standard edition is fully functional and is shipped on 5 CDs. There is no time limit for using the software. [*Note:* Professionals using this publication will have to purchase the necessary software to build and run the applications in this textbook.]

DIVE-INTO™ Series Tutorial for Visual C++ .NET

Our *DIVE-INTO™ SERIES* of tutorials helps readers get started with many popular program-development environments. These are available free for download at www.deitel.com/books/downloads.html. *DIVE-INTO Microsoft® Visual C++ .NET 2003* shows how to compile, execute and debug Visual C++ .NET applications in Visual Studio .NET Standard Edition version 2003. The document also provide step-by-step instructions with screen shots to help readers install the software, and overviews the compiler and its online documentation.

Ancillary Package for Visual C++ .NET How to Program

Visual C++ .NET How to Program has extensive ancillary materials for instructors. The *Instructor’s Resource CD (IRCD)* contains solutions to most of the end-of-chapter exercises. This CD is available only to instructors through their Prentice Hall representatives. [**NOTE: Please do not write to us requesting the instructor’s CD. Distribution of this CD is limited strictly to college professors teaching from the book. Instructors may obtain the solutions manual only from their Prentice Hall representatives.**] The ancillaries for this book also include a *Test Item File* of multiple-choice questions. In addition, we provide PowerPoint slides containing all the code and figures in the text and bulleted items that summarize the key points in the text. Instructors can customize the slides. The PowerPoint slides are downloadable from www.deitel.com and are available as part of Prentice Hall’s Companion Web Site (www.prenhall.com/deitel) for *Visual C++ .NET How to Program*, which offers resources for both instructors and students. For instructors, the Companion Web Site provides a Syllabus Manager, which helps instructors plan courses interactively and create online syllabi.

Students also benefit from the functionality of the *Companion Web Site*. Book-specific resources for students include:

- Customizable PowerPoint slides
- Source code for all example programs
- Reference materials from the book appendices (such as operator-precedence chart, character set and Web resources)

Chapter-specific resources available for students include:

- Chapter objectives
- Highlights (e.g., chapter summary)

- Outline
- Tips (e.g., *Common Programming Errors*, *Good Programming Practices*, *Portability Tips*, *Performance Tips*, *Look-and-Feel Observations*, *Software Engineering Observations* and *Error-Prevention Tips*)
- Online Study Guide—contains additional short-answer self-review exercises (e.g., true/false and matching questions) with answers and provides immediate feedback to the student

Students can track their results and course performance on quizzes using the *Student Profile* feature, which records and manages all feedback and results from tests taken on the *Companion Web Site*. To access the DEITEL[®] *Companion Web Sites*, visit www.prenhall.com/deitel.

DEITEL[®] e-Learning Initiatives

e-Books and Support for Wireless Devices

Wireless devices will have an enormous role in the future of the Internet. Given recent bandwidth enhancements and the emergence of 2.5 and 3G technologies, it is projected that, within a few years, more people will access the Internet through wireless devices than through desktop computers. Deitel & Associates is committed to wireless accessibility and has published *Wireless Internet & Mobile Business How to Program*. We are investigating new electronic formats, such as wireless e-books so that students and professors can access content virtually anytime, anywhere. For periodic updates on these initiatives subscribe to the DEITEL[®] *Buzz Online* e-mail newsletter, www.deitel.com/newsletter/subscribe.html or visit www.deitel.com.

DEITEL[®] Buzz Online E-mail Newsletter

Our free e-mail newsletter, the DEITEL[®] *Buzz Online*, includes commentary on industry trends and developments, links to free articles and resources from our published books and upcoming publications, product-release schedules, errata, challenges, anecdotes, information on our corporate instructor-led training courses and more. To subscribe, visit

www.deitel.com/newsletter/subscribe.html

Tour of the Book

In this section, we tour the chapters and appendices of *Visual C++ .NET How to Program*. In addition to the topics presented in each chapter, several of the chapters contain an Internet and Web Resources section that lists additional sources from which readers can enhance their knowledge of Visual C++ programming.

Chapter 1—Introduction to .NET and Visual C++[®] .NET

The first chapter presents the history of the Internet, World Wide Web and various technologies (such as XML and SOAP) that have led to advances in computing. We introduce the Microsoft .NET initiative and Visual C++ .NET, including Web services. We explore the impact of .NET on software development and software reusability.

Chapter 2—Introduction to the Visual Studio® .NET IDE

Chapter 2 introduces Visual Studio .NET, an integrated development environment (IDE) that allows programmers to create applications using standard C++ and Managed Extensions for C++ (Visual C++ .NET). Visual Studio .NET contains tools for debugging and writing code. The chapter presents features of Visual Studio .NET, including its key windows, and shows how to compile and run programs. The chapter also introduces readers to console-application and Windows-application programming in Visual C++ .NET. Every concept is presented in the context of a complete working Visual C++ .NET program and is followed by one or more screen shots showing actual inputs and outputs as the program executes.

Chapter 3—Introduction to Visual C++ .NET Programming

This chapter introduces readers to our LIVE-CODE approach. Every concept is presented in the context of a complete working Visual C++ .NET program and is followed by one or more sample outputs depicting the program's execution. In our first example, we print a line of text and carefully discuss each line of code. We then discuss fundamental tasks, such as how a program inputs data from its users and how to write programs that perform arithmetic.

Chapter 4—Control Statements: Part 1

This chapter formally introduces the principles of structured programming, a set of techniques that will help the reader develop clear, understandable, maintainable programs throughout the text. The first part of this chapter presents program-development and problem-solving techniques. The chapter demonstrates how to transform a written specification to a program by using such techniques as *pseudocode* and *top-down, stepwise refinement*. We then progress through the entire process, from developing a problem statement into a working Visual C++ .NET program. The notion of algorithms is also discussed. We build on information presented in the previous chapter to create programs that are interactive (i.e., they change their behavior to suit user-supplied inputs). The chapter then introduces the use of control statements that affect the sequence in which statements are executed. Control statements produce programs that are easily understood, debugged and maintained. We discuss the three forms of program control—sequence, selection and repetition—focusing on the `if...else` and `while` control statements. Flowcharts (i.e., graphical representations of algorithms) appear throughout the chapter, reinforcing and augmenting the explanations.

Chapter 5—Control Statements: Part 2

Chapter 5 introduces more complex control statements and the logical operators. It uses flowcharts to illustrate the flow of control through each control structure, including the `for`, `do...while` and `switch` statements. We explain the `break` and `continue` statements and the logical operators. Examples include calculating compound interest and printing the distribution of grades on an exam (with some simple error checking). The chapter concludes with a structured programming summary, including each of Visual C++ .NET's control statements. The techniques discussed in Chapters 4 and 5 constitute a large part of what has been taught traditionally under the topic of structured programming.

Chapter 6—Functions

A *function* allows the programmer to create a block of code that can be called upon from various points in a program. Groups of related functions can be separated into functional blocks

(classes), using the “divide and conquer” strategy. Programs are divided into simple components that interact in straightforward ways. We discuss how to create our own functions that can take input, perform calculations and return output. We examine the FCL’s `Math` class, which contains methods for performing complex calculations (e.g., trigonometric and logarithmic calculations). The FCL (Framework Class Library) is .NET’s code library, or collection of classes that provides numerous capabilities to the programmer. *Recursive* functions (functions that call themselves) and function overloading, which allows multiple functions to have the same name, are introduced. We demonstrate overloading by creating two `Square` functions that take an integer (i.e., whole number) and a floating-point number (i.e., a number with a decimal point), respectively.

Chapter 7—Arrays

Chapter 7 discusses arrays, our first data structures. (Chapter 22 discusses the topic of data structures in depth.) Data structures are crucial to storing, sorting, searching and manipulating large amounts of information. *Arrays* are groups of related data items that allow the programmer to access any element directly. Rather than creating 100 separate variables that are all related in some way, the programmer instead can create an array of 100 elements and access these elements by their location in the array. We discuss how to declare and allocate managed arrays, and we build on the techniques of the previous chapter by passing arrays to functions. In addition, we discuss how to pass a variable number of arguments to methods. Chapters 4 and 5 are essential for array processing, because repetition statements are used to iterate through elements in an array. The combination of these concepts helps the reader create highly-structured and well-organized programs. We then demonstrate how to sort and search arrays. We discuss multidimensional arrays, which can be used to store tables of data.

Chapter 8—Object-Based Programming

Chapter 8 begins our deeper discussion of classes. The chapter represents a wonderful opportunity for teaching data abstraction the “right way”—through a language (MC++) expressly devoted to implementing new types. The chapter focuses on the essence and terminology of classes (programmer-defined types) and objects. The chapter discusses implementing MC++ classes, accessing class members, enforcing information hiding with access modifiers, separating interface from implementation, using properties and utility methods and initializing objects with constructors. The chapter discusses declaring and using constants, *composition*, the `this` reference, `static` class members and examples of popular abstract data types such as stacks and queues. We overview how to create reusable software components with assemblies, namespaces and Dynamic Link Library (DLL) files.

Chapter 9—Object-Oriented Programming: Inheritance

Chapter 9 introduces one of the most fundamental capabilities of object-oriented programming languages, inheritance, which is a form of software reusability in which new classes are developed quickly and easily by absorbing the capabilities of existing classes and adding appropriate new capabilities. The chapter discusses the notions of base classes and derived classes, access modifier `protected`, direct base classes, indirect base classes, use of constructors in base classes and derived classes, and software engineering with inheritance. The chapter compares inheritance (“is a” relationships) with composition (“has a” relationships).

Chapter 10—Object-Oriented Programming: Polymorphism

Chapter 10 deals with another fundamental capability of object-oriented programming, namely polymorphic behavior. *Polymorphism* permits classes to be treated in a general manner, allowing the same method call to act differently depending on context (e.g., “move” messages sent to a bird and a fish result in dramatically different types of action—a bird flies and a fish swims). In addition to treating existing classes in a general manner, polymorphism allows new classes to be added to a system easily. This chapter distinguishes between abstract classes and concrete classes. A feature of this chapter is its three polymorphism case studies—a payroll system, a shape hierarchy headed up by an abstract class and a shape hierarchy headed up by an interface. These programming techniques and those of the previous chapter allow the programmer to create extensible and reusable software components.

Chapter 11—Exception Handling

Exception handling is one of the most important topics in Visual C++ .NET from the standpoint of building mission-critical and business-critical applications. People can enter incorrect data, data can be corrupted and clients can try to access records that do not exist or are restricted. A simple division-by-zero error may cause a calculator program to crash, but what if such an error occurs in the navigation system of a flying airplane? Programmers must deal with these situations, because in some cases, the results of program failure could be disastrous. Programmers need to know how to recognize the errors (*exceptions*) that might occur in software components and handle those exceptions effectively, allowing programs to deal with problems and continue executing instead of “crashing.” This chapter overviews exception-handling techniques. We cover the details of Visual C++ .NET exception handling, the termination model of exception handling, throwing and catching exceptions, and the FCL class `Exception`. Programmers who construct software systems from reusable components built by other programmers often deal with the exceptions that those components throw when a problem occurs.

Chapter 12—Graphical User Interface Concepts: Part 1

Chapter 12 explains how to add graphical user interfaces (GUIs) to our programs, providing a professional look and feel. By using the techniques of rapid application development (RAD), we can create a GUI from reusable controls, rather than explicitly programming every detail. The Visual Studio .NET IDE makes developing GUIs even easier by allowing the programmer to position components in a window through so-called visual programming. We discuss how to construct user interfaces with *Windows Forms* GUI controls such as labels, buttons, text boxes, scroll bars and picture boxes. We also introduce *events*, which are messages sent by a program to signal to an object or a set of objects that an action has occurred. Events are most commonly used to signal user interactions with GUI controls, but also can signal internal actions in a program. We overview event handling and discuss how to handle events specific to controls, the keyboard and the mouse. Tips are included throughout the chapter to help the programmer create visually appealing, well-organized and consistent GUIs.

Chapter 13—Graphical User Interface Concepts: Part 2

Chapter 13 introduces more complex GUI controls, including menus, link labels, panels, list boxes, combo boxes and tab controls. In a challenging exercise, readers create an application that displays a disk drive’s directory structure in a tree—similar to that created by

Windows Explorer. The *Multiple Document Interface (MDI)* is presented, which allows multiple windows to be open simultaneously in a single GUI. We conclude with a discussion of how to create custom controls by combining existing controls. The techniques presented in this chapter allow readers to create sophisticated and well-organized GUIs, adding style and usability to their applications.

Chapter 14—Multithreading

We have come to expect much from our applications. We want to download files from the Internet, listen to music, print documents and browse the Web—all at the same time! To do this, we need a technique called multithreading, which allows applications to perform multiple activities concurrently. Visual C++ .NET gives programmers access to the multithreading classes provided by the FCL, while shielding programmers from complex details. Visual C++ .NET is better equipped to deal with more sophisticated multimedia, network-based and multiprocessor-based applications than other languages that do not have multithreading features. This chapter overviews the threading classes in the FCL and covers threads, thread life-cycles, time-slicing, scheduling and priorities. We analyze the producer-consumer relationship, thread synchronization and circular buffers. This chapter lays the foundation for creating the impressive multithreaded programs that clients demand.

Chapter 15—Strings, Characters and Regular Expressions

In this chapter, we discuss the processing of words, sentences, characters and groups of characters. In Visual C++ .NET, *strings* (groups of characters) are objects. This is yet another benefit of Visual C++ .NET's emphasis on object-oriented programming. *String* objects contain methods that can copy, search, extract substrings and concatenate strings with one another. We introduce class *StringBuilder*, which defines string-like objects that can be modified after initialization. As an interesting example of strings, we create a card shuffling-and-dealing simulation. We discuss regular expressions, a powerful tool for searching and manipulating text.

Chapter 16—Graphics and Multimedia

In this chapter, we discuss *GDI+* (an extension of the *Graphics Device Interface—GDI*), the Windows service that provides the graphical features used by .NET applications. The extensive graphical capabilities of GDI+ can make programs more visual and fun to create and use. We discuss Visual C++ .NET's treatment of graphics objects and color control. We also discuss how to draw arcs, polygons and other shapes. The chapter demonstrates how to use various pens and brushes to create color effects and includes an example that demonstrates gradient fills and textures. We also introduce techniques for turning text-only applications into aesthetically pleasing programs that even novice programmers can write with ease. The second half of the chapter focuses on audio, video and speech technology. We discuss adding sound, video and animated characters to programs (primarily via existing audio and video clips). You will see how easy it is to incorporate multimedia into Visual C++ .NET applications. This chapter introduces a technology called *Microsoft Agent* for adding *interactive animated characters* to a program. Each character allows users to interact with the application, using more natural human communication techniques, such as speech. The agent characters respond to mouse and keyboard events, speak and hear (i.e., they support speech synthesis and speech recognition). With these capabilities, your applications can speak to users and actually respond to their voice commands!

Chapter 17—Files and Streams

Imagine a program that could not save data to a file. Once the program is closed, all the work performed by the program is lost forever. For this reason, this chapter is one of the most important for programmers who will be developing commercial applications. We introduce FCL classes for inputting and outputting data. A detailed example demonstrates these concepts by allowing users to read and write bank account information to and from files. We introduce the FCL classes and methods that help perform input and output conveniently—they demonstrate the power of object-oriented programming and reusable classes. We discuss benefits of sequential files, random-access files and buffering. This chapter lays the groundwork for the material presented in Chapter 21, Networking: Streams-Based Sockets and Datagrams.

Chapter 18—Extensible Markup Language (XML)

The Extensible Markup Language (XML) derives from SGML (Standard Generalized Markup Language), which became an industry standard in 1986. Although SGML is employed in publishing applications worldwide, it has not been incorporated into the mainstream programming community because of its sheer size and complexity. XML is an effort to make SGML-like technology available to a much broader community. XML, created by the World Wide Web Consortium (W3C), describes data in a portable format. XML differs in concept from markup languages such as HTML, which only describes how information is rendered in a browser. XML is a technology for creating markup languages for virtually any type of information. Document authors use XML to create entirely new markup languages to describe specific types of data, including mathematical formulas, chemical molecular structures, music, recipes and much more. Markup languages created with XML include XHTML (Extensible HyperText Markup Language, for Web content), MathML (for mathematics), VoiceXML™ (for speech), SMIL™ (Synchronized Multimedia Integration Language, for multimedia presentations), CML (Chemical Markup Language, for chemistry) and XBRL (Extensible Business Reporting Language, for financial data exchange). The extensibility of XML has made it one of the most important technologies in industry today and is being integrated into almost every field. Companies and individuals constantly are finding new and innovative uses for XML. In this chapter, we present examples that illustrate the basics of marking up data with XML. We demonstrate several XML-derived markup languages, such as *XML Schema* (for checking an XML document's grammar), and *XSLT (Extensible Stylesheet Language Transformations)*, for transforming an XML document's data into another text-based format such as XHTML). (For readers who are unfamiliar with XHTML, we provide Appendices E and F, which present a detailed introduction to XHTML.)

Chapter 19—Database, SQL and ADO .NET

Data storage and access are integral to creating powerful software applications. This chapter discusses .NET support for database manipulation. Today's most popular database systems are relational databases. In this chapter, we introduce the *Structured Query Language (SQL)* for performing queries on relational databases. We also introduce *ActiveX Data Objects (ADO .NET)*—an extension of ADO that enables .NET applications to access and manipulate databases. ADO .NET allows data to be exported as XML, which enables ADO .NET applications to communicate with programs that understand XML. The reader will learn how to create database connections, using tools provided in Visual Studio .NET, and how to use ADO .NET classes to query a database.

Chapter 20—Web Services

Previous chapters demonstrated how to create applications that execute locally on the user's computer. In this chapter, we introduce Web services, which are programs that “expose” services (i.e., methods) to clients over the Internet, intranets and extranets. Web services offer increased software reusability by allowing services on disparate platforms to interact with each other seamlessly. We discuss .NET Web services basics and related technologies, including *Simple Object Access Protocol (SOAP)* and *Active Server Pages (ASP)* .NET. This chapter presents an interesting example of a Web service that manipulates huge integers (up to 100 digits). We present a Blackjack application that demonstrates session tracking, a form of personalization that enables the application to “recognize” a user. We conclude with a discussion of Microsoft's Global XML Web Services Architecture (GXA), a series of specifications that provide additional capabilities to Web services developers.

Chapter 21—Networking: Streams-Based Sockets and Datagrams

Chapter 21 introduces the fundamental techniques of streams-based networking. We demonstrate how streams-based *sockets* allow programmers to hide many networking details. With sockets, networking is as simple as if the programmer were reading from and writing to a file. We also introduce *datagrams*, in which packets of information are sent between programs. Each packet is addressed to its recipient and sent out to the network, which routes the packet to its destination. The examples in this chapter focus on communication between applications. One example demonstrates using streams-based sockets to communicate between two Visual C++ .NET programs. Another, similar example sends datagrams between applications. We also show how to create a multithreaded-server application that can communicate with multiple clients in parallel. In this client/server tic-tac-toe game, the server maintains the status of the game, and two clients communicate with the server to play the game.

Chapter 22—Data Structures and Collections

This chapter discusses arranging data into aggregations such as linked lists, stacks, queues and trees. Each data structure has properties that are useful in many applications, from sorting elements to keeping track of method calls. We discuss how to build each of these data structures. This is also a valuable experience in crafting useful classes. In addition, we cover pre-built collection classes in the FCL. These classes store sets, or collections, of data and provide functionality that allow the developer to sort, insert, delete and retrieve data items. Different collection classes store data in different ways. This chapter focuses on classes `Array`, `ArrayList`, `Stack` and `Hashtable`, discussing the details of each. When possible, Visual C++ .NET programmers should use appropriate FCL collections, rather than implementing similar data structures themselves. This chapter reinforces much of the object technology discussed in Chapters 5–7, including classes, inheritance and composition.

Appendix A—Operator Precedence Chart

This appendix lists Visual C++ .NET operators and their precedence.

Appendix B—Number Systems

This appendix explains the binary, octal, decimal and hexadecimal number systems. It also reviews the conversion of numbers among these bases and illustrates mathematical operations in each base.

Appendix C—ASCII Character Set

This appendix contains a table of the 128 ASCII (American Standard Code for Information Interchange) alphanumeric symbols and their corresponding integer values.

Appendix D—Unicode®

This appendix introduces the Unicode Standard, an encoding scheme that assigns unique numeric values to the characters of most of the world's languages. We include a Windows application that uses Unicode encoding to print welcome messages in several languages.

Appendices E and F—Introduction to XHTML: Parts 1 & 2

In these appendices, we introduce the Extensible HyperText Markup Language (XHTML), a W3C technology designed to replace HTML as the primary means of describing Web content. As an XML-based language, XHTML is more robust and extensible than HTML. XHTML incorporates most of HTML's elements and attributes—the focus of these appendices. Appendices E and F are included for our readers who do not know XHTML or who would like a review of XHTML before studying Chapter 18, Extensible Markup Language (XML).

Appendix G—XHTML Special Characters

This appendix provides many commonly used XHTML special characters, called *character entity references*.

Appendix H—XHTML Colors

This appendix lists commonly used XHTML color names and their corresponding hexadecimal values.

Appendix I—Bit Manipulation

This appendix discusses Visual C++ .NET's powerful bit-manipulation capabilities. This helps programs process bit strings, set individual bits on or off and store information more compactly. Such capabilities are characteristic of low-level assembly languages and are valued by programmers writing systems software, such as operating system and networking software.

Acknowledgments

One of the great pleasures of writing a textbook is acknowledging the efforts of many people whose names may not appear on the cover, but whose hard work, cooperation, friendship and understanding were crucial to the production of the book. Many people at Deitel & Associates, Inc. devoted long hours to this project:

Abbey Deitel
Barbara Deitel
Christi Kelsey
Tem Nieto
Christina Courtemarche
Rashmi Jayaprakash
Laura Treibick
Betsy DuWaldt

We would also like to thank the participants in the Deitel & Associates, Inc., College Internship Program who contributed to this publication.¹ We would like to extend a special thank you to Jim Bai of Carnegie Mellon University, who helped us with the completion of the text and instructor's manual under tight deadlines.

Jim Bai (Carnegie Mellon)
Bei Zhao (Northeastern)
Jimmy Nguyen (Northeastern)
Nicholas Cassie (Northeastern)
Thiago da Silva (Northeastern)
Mike Dos'Santos (Northeastern)
Emanuel Achildiev (Northeastern)

We are fortunate to have worked on this project with the talented and dedicated team of publishing professionals at Prentice Hall. We especially appreciate the extraordinary efforts of our Computer Science Editor, Kate Hargett and her boss and our mentor in publishing—Marcia Horton, Editorial Director of Prentice-Hall's Engineering and Computer Science Division. Vince O'Brien and Tom Manshreck did a marvelous job managing the production of the book. Sarah Parker managed the publication of the book's extensive ancillary package.

We wish to acknowledge the efforts of our reviewers and to thank Carole Snyder and Jennifer Cappello of Prentice Hall, who managed the review process. Adhering to a tight time schedule, these reviewers scrutinized the text and the programs, providing countless suggestions for improving the accuracy and completeness of the presentation. We sincerely appreciate the time these people took from their busy professional schedules to help us ensure the quality, accuracy and timeliness of this book.

Visual C++ .NET How to Program Reviewers:
Shishir Abhyanker (Accenture)
Rekha Bhowmik (Winona State University)
Chadi Boudiab (Georgia Perimeter College)
Steve Chattaroon (Northern Alberta Institute of Technology)
Kunal Cheda (Syntel India, Ltd.)
Dean Goodmanson (Renaissance Learning, Inc.)
Keith Harrow (Brooklyn College)
Doug Harrison (Eluent Software)
James Huddleston (Independent Consultant)
Terrell Hull (Sun Certified Java Architect, Rational Qualified Practitioner)
Shrawan Kumar (Accenture)
Andrew Mooney (American Continental University)

-
1. The *Deitel & Associates, Inc. College Internship Program* offers a limited number of salaried positions to Boston-area college students majoring in Computer Science, Information Technology, Marketing, Management and English. Students work at our corporate headquarters in Maynard, Massachusetts full-time in the summers and (for those attending college in the Boston area) part-time during the academic year. We also offer full-time internship positions for students interested in taking a semester off from school to gain industry experience. Regular full-time positions are available from time to time to college graduates. For more information, please contact our president—abbey.deitel@deitel.com—and visit our Web site, www.deitel.com.

Neal Patel (Microsoft Corporation)
Paul Randal (Microsoft Corporation)
Christopher Whitehead (Columbus State University)
Warren Wiltsie (Fairleigh Dickinson University)

Visual C++ .NET: A Managed Code Approach for Experienced Programmers

Reviewers:

Neal Patel (Microsoft)
Paul Randal (Microsoft)
Scott Woodgate (Microsoft)
David Weller (Microsoft)
Dr. Rekha Bhowmik (St. Cloud State University)
Carl Burnham (Hosting Resolve)
Kyle Gabhart (StarMaker Technologies)
Doug Harrison (Eluent Software)
Christian Hessler (Sun Microsystems)
Michael Hudson (Blue Print Tech)
John Paul Mueller (DataCon Services)
Nicholas Paldino (Exis Consulting)
Chris Platt (RealAge Inc./ UC San Diego Extension)
Teri Radichel (Radical Software)
Ivan Rancati
Tomas Restrepo (Intergrupo S.A)

Contacting Deitel & Associates

We would sincerely appreciate your comments, criticisms, corrections and suggestions for improving the text. Please address all correspondence to:

deitel@deitel.com

We will respond promptly.

Errata

We will post all errata for this publication at www.deitel.com.

Customer Support

Please direct all software and installation questions to Pearson Education Technical Support:

- By phone: 1-800-677-6337
- By email: media.support@pearsoned.com
- On the Web: 247.prenhall.com

Please direct all Visual C++ .NET language questions to deitel@deitel.com. We will respond promptly.

Welcome to the exciting world of programming in Visual C++ .NET. We sincerely hope you enjoy learning with this book.

Dr. Harvey M. Deitel
Paul J. Deitel
Jonathan P. Liperi
Cheryl H. Yaeger

About the Authors

Dr. Harvey M. Deitel, Chairman of Deitel & Associates, Inc., has 42 years experience in the computing field, including extensive industry and academic experience. Dr. Deitel earned B.S. and M.S. degrees from the Massachusetts Institute of Technology and a Ph.D. from Boston University. He worked on the pioneering virtual memory operating-systems projects at IBM and MIT that developed techniques now widely implemented in systems such as UNIX, Linux and Windows XP. He has 20 years of college teaching experience and served as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., with his son, Paul J. Deitel. He is the author or co-author of several dozen books and multimedia packages. With translations published in numerous foreign languages, Dr. Deitel's texts have earned international recognition. Dr. Deitel has delivered professional seminars to major corporations, government organizations and various branches of the military.

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of the Massachusetts Institute of Technology's Sloan School of Management, where he studied information technology. Through Deitel & Associates, Inc., he has delivered professional seminars to numerous industry and government clients and has lectured on C++ and Java for the Boston Chapter of the Association for Computing Machinery. He and his father, Dr. Harvey M. Deitel, are the world's best-selling Computer Science textbook authors.

Jonathan P. Liperi is a graduate of Boston University with a Master's degree in Computer Science. His research at BU focused on dimensionality reduction techniques and matching functions for temporal databases. Jon also co-authored Deitel & Associates, Inc. publications, *Visual C++ .NET: A Managed Code Approach for Experienced Programmers* and *Python How to Program*. Jon currently works as a Software Development Engineer in Test on the Enterprise Frameworks and Tools team at Microsoft.

Cheryl H. Yaeger, Director of Microsoft Software Publications with Deitel & Associates, Inc., is a graduate of Boston University with a degree in Computer Science. Cheryl has co-authored various Deitel & Associates, Inc. publications, including *Visual C++ .NET: A Managed Code Approach for Experienced Programmers*, *C# How to Program*, *C#: A Programmer's Introduction*, *C# for Experienced Programmers*, *Simply C#*, *Visual Basic .NET for Experienced Programmers*, *Simply Visual Basic .NET*, *Simply Visual Basic .NET 2003* and *Simply Java™ Programming* and has contributed to several others.

About Deitel & Associates, Inc.

Deitel & Associates, Inc., is an internationally recognized corporate training and content-creation organization specializing in Internet/World Wide Web software technology, e-business/e-commerce software technology, object technology and computer programming languages education. The company provides instructor-led courses on Internet and World Wide Web programming, wireless Internet programming, object technology, and major programming languages and platforms, such as C, C++, Visual C++ .NET, Visual Basic .NET, C#, Java, Advanced Java, XML, Perl, Python and more. The founders of Deitel & Associates, Inc., are Dr. Harvey M. Deitel and Paul J. Deitel. The company's clients include many of the world's largest computer companies, government agencies, branches of the military and business organizations. Through its 28-year publishing partnership with

Prentice Hall, Deitel & Associates, Inc., publishes leading-edge programming textbooks, professional books, interactive CD-based multimedia *Cyber Classrooms*, *Complete Training Courses*, Web-based training courses and course management systems e-content for popular CMSs such as WebCT™, Blackboard™ and CourseCompassSM. Deitel & Associates, Inc., and the authors can be reached via e-mail at:

deitel@deitel.com

To learn more about Deitel & Associates, Inc., its publications and its worldwide corporate on-site training curriculum, see the last few pages of this book or visit:

www.deitel.com

Individuals wishing to purchase Deitel books, *Cyber Classrooms*, *Complete Training Courses* and Web-based training courses can do so through bookstores, online booksellers and:

www.deitel.com
www.prenhall.com/deitel
www.InformIT.com/deitel
www.InformIT.com/cyberclassrooms

Bulk orders by corporations and academic institutions should be placed directly with Prentice Hall. See the last few pages of this book for worldwide ordering instructions.