Preface

"The chief merit of language is clearness ..."
—Galen

Welcome to Visual C++ 2008 and *Visual C++ 2008 How to Program, Second Edition*! At Deitel & Associates, we write programming language textbooks and professional books for Prentice Hall, deliver corporate training courses worldwide and develop Web 2.0 Internet businesses. This book reflects significant changes to the Visual C++ language and to the preferred ways of teaching and learning programming. The book is based on our C++-standard-compliant textbook *C++ How to Program, Sixth Edition* and is intended for courses that offer a Microsoft-specific C++ programming focus.

Since the first edition of *Visual C++ How to Program*, Microsoft has changed the way C++ interacts with the .NET Framework—the Managed Extensions to C++ have been replaced by the cleaner C++/CLI. Deitel has also refined the pedagogy used in its books since the first edition of *Visual C++ How to Program*. Consequently, all the chapters have undergone significant updates and tuning. If you're interested in learning Visual C++ using Microsoft's Visual Studio Integrated Development Environment (IDE) and the .NET Framework, then this book is for you.

New and Updated Features

Here's the updates we've made to the second edition of *Visual C++ 2008 How to Program*:

- **Early Classes and Objects Approach.** Students are introduced to the basic concepts and terminology of object technology in Chapter 1 and begin developing customized, reusable classes and objects in Chapter 4, using native C++ and managed code with C++/CLI. This book presents object-oriented programming, where appropriate, from the start and throughout the text. The early discussion of objects and classes gets students "thinking about objects" immediately and mastering these concepts more completely. Object-oriented programming is not trivial by any means, but it's fun to write object-oriented programs, and students can see immediate results.
- **New Native-Code Approach.** Microsoft has determined that most Visual C++ developers do the majority of their programming in native C++. As a result, this edition of the book represents a major overhaul in approach from the first edition. We now introduce new programming concepts first with native C++ followed by managed code sections with C++/CLI, where appropriate. We worked closely with members of Microsoft's Visual C++ team and determined that this was the best approach for this new edition.

- **Major Content Revisions and Updates for the .NET Framework.** All the chapters have been significantly updated and upgraded. We added sections and chapters introducing managed code concepts with C++/CLI. We tuned the writing for clarity and precision and adjusted our use of Visual C++ terminology in accordance with the ISO/IEC standard document that defines the C++ language.
- **Introduction to the Visual C++ 2008 Express IDE and the Visual Studio Debugger.** Chapter 2 provides a detailed tutorial on using the Visual C++ Express 2008 Integrated Development Environment to create and run native C++ and .NET projects. Appendix H, explains the basics of debugging programs using Visual C++ Express 2008. You can download Visual C++ 2008 Express from `www.microsoft.com/express/vc/`.
- **The Managed Heap and CLR Garbage Collector.** Chapters 9 and 11 include new sections on the managed heap, the CLR garbage collector, and memory management in .NET. We introduce handles and tracking references for using managed objects with C++/CLI.
- **New Content on the .NET Framework Class Library (FCL).** New sections introducing managed code concepts with C++/CLI use numerous classes from the .NET Framework Class Library. Chapter 18 teaches file processing in .NET using the `File` and `Directory` classes. Chapter 19 provides an in-depth look at classes `String`, `StringBuilder` and `Char`. Chapter 25 focuses exclusively on collections in the .NET Framework using FCL classes.
- **Exception Handling in .NET.** After introducing exception handling in native C++, Chapter 16 introduces the .NET `Exception` class hierarchy. We demonstrate the use of `finally` and stack semantics for writing safer C++/CLI code.
- **Templates and Generics.** New sections in Chapter 15 introduce managed templates in C++/CLI as well as .NET generics. We compare and contrast the strengths and weaknesses of both types of generic programming.
- **Integrated Case Studies.** We provide several case studies spanning multiple sections and chapters that often build on a class introduced earlier in the book to demonstrate new programming concepts later in the book. These case studies include the development of a `GradeBook` class in Chapters 4–8, a `Time` class in several sections of Chapters 10–11, an `Employee` class in Chapters 13–14, and the optional OOD/UML ATM case study in Chapters 1, 3–8, 10, 14 and Appendix F.
- **Integrated `GradeBook` Case Study.** The `GradeBook` case study reinforces our early classes presentation. It uses classes and objects in Chapters 4–8 to incrementally build a `GradeBook` class that represents an instructor's grade book and performs various calculations based on a set of student grades, such as calculating the average grade, finding the maximum and minimum, and printing a bar chart.
- **Unified Modeling Language™ 2 (UML 2).** The Unified Modeling Language (UML) has become the preferred graphical modeling language for designers of object-oriented systems. All the UML diagrams in the book comply with the UML 2 specification. We use UML class diagrams to visually represent classes and their inheritance relationships, and we use UML activity diagrams to dem-

onstrate the flow of control in each of Visual C++'s control statements. We use the UML extensively in the optional OOD/UML ATM case study.

- **Optional OOD/UML ATM Case Study.** The optional OOD/UML automated teller machine (ATM) case study in the Software Engineering Case Study sections of Chapters 1, 3–8, 10 and 14 is appropriate for first and second programming courses. The case study sections present a carefully paced introduction to object-oriented design using the UML. We introduce a concise, simplified subset of the UML 2, then guide you through a first design experience intended for the novice object-oriented designer/programmer. Our goal in this case study is to help students develop an object-oriented design to complement the object-oriented programming concepts they begin learning in Chapter 1 and implementing in Chapter 4. The case study was reviewed by a distinguished team of OOD/UML academic and industry professionals. The case study is not an exercise; rather, it is a fully developed end-to-end learning experience that concludes with a detailed walkthrough of the complete 877-line C++ code implementation. We take a detailed tour of the case study later in the Preface.
- **Compilation and Linking Process for Multiple-Source-File Programs.** Chapter 4 includes a detailed diagram and discussion of the compilation and linking process that produces an executable application.
- **Function Call Stack Explanation.** In Chapter 7, we provide a detailed discussion (with illustrations) of the function-call stack and activation records to explain how Visual C++ is able to keep track of which function is currently executing, how automatic variables of functions are maintained in memory, and how a function knows where to return after it completes execution.
- **C++ Standard Library `string` and `vector` Classes.** The `string` and `vector` classes are used to make earlier examples more object-oriented.
- **Class `string`.** We use class `string` instead of C-like pointer-based `char *` strings for most string manipulations throughout the book. We continue to include discussions of `char *` strings in Chapters 9, 11, 12 and 22 to give students practice with pointer manipulations, to illustrate dynamic memory allocation with `new` and `delete`, to build our own `String` class, and to prepare students for working with `char *` strings in C and C++ legacy code.
- **Class Template `vector`.** We use class template `vector` instead of C-like pointer-based array manipulations throughout the book. However, we begin by discussing C-like pointer-based arrays in Chapter 8 to prepare students for working with C and C++ legacy code and to establish a basis for building our own customized `Array` class in Chapter 12.
- **Tuned Treatment of Inheritance and Polymorphism.** Chapters 13–14 have been carefully tuned using an `Employee` class hierarchy to make the treatment of inheritance and polymorphism clearer, more natural and more accessible for students who are new to OOP.
- **Discussion and Illustration of How Polymorphism Works "Under the Hood."** Chapter 14 contains a detailed diagram and explanation of how Visual C++ can

implement polymorphism, `virtual` functions and dynamic binding internally. This gives students a solid understanding of how these capabilities really work. More importantly, it helps students appreciate the overhead of polymorphism—in terms of additional memory consumption and processor time. This helps students determine when to use polymorphism and when to avoid it.

- **Standard Template Library (STL).** This might be one of the most important topics in the book in terms of your appreciation of software reuse. The STL defines powerful, template-based, reusable components that implement many common data structures and algorithms used to process those data structures. Chapter 23 introduces the STL and discusses its three key components—containers, iterators and algorithms. We show that using STL components provides tremendous expressive power, often reducing many lines of code to a single statement. We also introduce the STL/CLR, a new Microsoft library enabling managed code with C++/CLI to leverage the power of STL containers and algorithms.
- **ISO/IEC C++ Standard Compliance.** We have audited our presentation against the most recent ISO/IEC C++ standard document for completeness and accuracy. [*Note:* A PDF copy of the C++ standard (document number INCITS/ISO/IEC 14882-2003) can be purchased at `webstore.ansi.org/ansidocstore/default.asp`.]

All of this has been carefully reviewed by distinguished academics and industry developers who worked with us on *C++ How to Program, Sixth Edition* and *Visual C++ 2008 How to Program*.

We believe that this book and its support materials will provide students and professionals with an informative, interesting, challenging and entertaining Visual C++ educational experience. We also provide a suite of ancillary materials that help instructors maximize their students' learning experience.

As you read the book, if you have questions, send an e-mail to `deitel@deitel.com`; we'll respond promptly. For updates on this book and the status of all supporting Visual C++ software, and for the latest news on all Deitel publications and services, visit `www.deitel.com`. Sign up at `www.deitel.com/newsletter/subscribe.html` for the free *Deitel® Buzz Online* e-mail newsletter and check out our growing list of Visual C++ and related Resource Centers at `www.deitel.com/ResourceCenters.html`. Each week we announce our latest Resource Centers in the newsletter. Please let us know of other Resource Centers you'd like to see.

Dependency Chart

Figure 1 illustrates the dependencies that exist between chapters in the book. An arrow pointing into a chapter indicates that the chapter depends on the content of the chapter from which the arrow points. We recommend that you study all of a given chapter's dependencies before studying that chapter, though other orders are possible. Some of the dependencies apply only to sections of chapters, so we advise readers to browse the material before designing a course of study. We've also commented on some additional dependencies in the diagram's footnotes. This book is intended for courses that prefer a more Microsoft-specific C++ focus and that mix and match some native C++ and some managed C++.

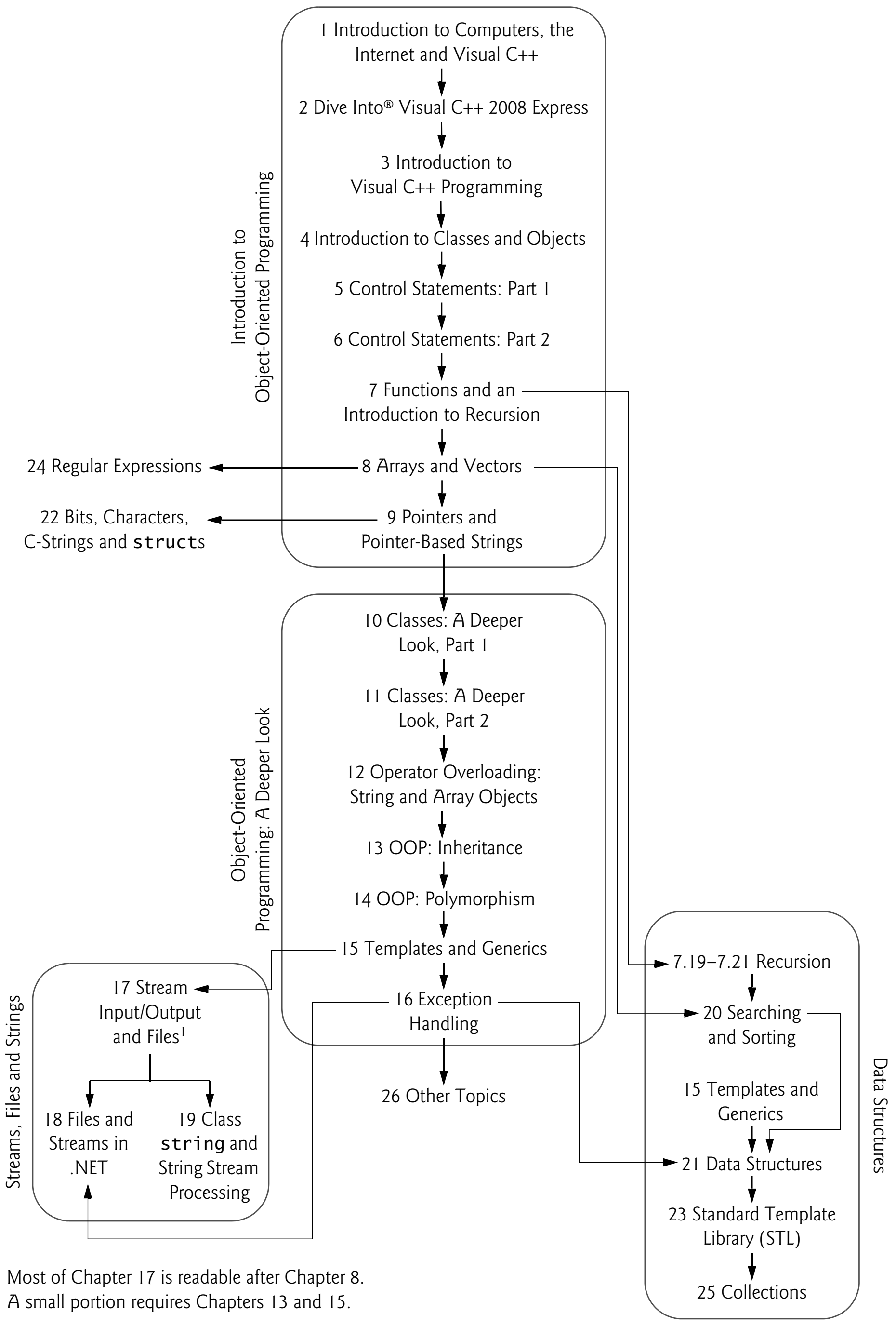
Introduction to Object-Oriented Programming
1 Introduction to Computers, the Internet and Visual C++
2 Dive Into® Visual C++ 2008 Express
3 Introduction to Visual C++ Programming
4 Introduction to Classes and Objects
5 Control Statements: Part 1
6 Control Statements: Part 2
7 Functions and an Introduction to Recursion
8 Arrays and Vectors
9 Pointers and Pointer-Based Strings
24 Regular Expressions
22 Bits, Characters, C-Strings and structs
Object-Oriented Programming: A Deeper Look
10 Classes: A Deeper Look, Part 1
11 Classes: A Deeper Look, Part 2
12 Operator Overloading: String and Array Objects
13 OOP: Inheritance
14 OOP: Polymorphism
15 Templates and Generics
16 Exception Handling
26 Other Topics
Streams, Files and Strings
17 Stream Input/Output and Files[1]
18 Files and Streams in .NET
19 Class string and String Stream Processing
Data Structures
7.19–7.21 Recursion
20 Searching and Sorting
15 Templates and Generics
21 Data Structures
23 Standard Template Library (STL)
25 Collections
[1] Most of Chapter 17 is readable after Chapter 8. A small portion requires Chapters 13 and 15.

Fig. 1 | *Visual C++ 2008 How to Program, 2/e* chapter dependency chart.

Teaching Approach

Visual C++ 2008 How to Program, 2/e contains a rich collection of examples. The book concentrates on the principles of good software engineering and stresses program clarity. We teach by example. We are educators who teach leading-edge topics in industry classrooms worldwide. Dr. Harvey M. Deitel has 20 years of college teaching experience and 18 years of industry teaching experience. Paul Deitel has 16 years of industry teaching experience. The Deitels have taught courses at all levels to government, industry, military and academic clients of Deitel & Associates.

Live-Code Approach. *Visual C++ 2008 How to Program, 2/e* is loaded with "live-code" examples—by this we mean that each new concept is presented in the context of a complete working Visual C++ program that is immediately followed by one or more actual executions showing the program's inputs and outputs. This style exemplifies the way we teach and write about programming; we call this the "live-code approach."

Syntax Shading. We syntax shade all the Visual C++ code, similar to the way Visual Studio syntax colors code. This improves code readability—an important goal, given that the book contains about 19,000 lines of code in complete, working Visual C++ programs. Our syntax-shading conventions are as follows:

```
comments appear in italic
keywords appear in bold italic
errors appear in bold black
constants and literal values appear in bold, gray
all other code appears in plain black
```

Code Highlighting. We place gray rectangles around the key code segments in each program.

Programming Tips. We include programming tips to help you focus on important aspects of program development. These tips and practices represent the best we have gleaned from a combined six decades of programming and teaching experience. One of our students—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 basis on which to build good software.

Good Programming Practice 1.1

Good Programming Practices *call attention to techniques that will help you produce programs that are clearer, more understandable and more maintainable.*

Common Programming Error 1.1

Students tend to make certain kinds of errors frequently. Pointing out these Common Programming Errors *reduces the likelihood that you'll make the same mistakes.*

Error-Prevention Tip 1.1

These tips contain suggestions for exposing bugs and removing them from your programs; many describe aspects of Visual C++ that prevent bugs from getting into programs in the first place.

Performance Tip 1.1

Students like to "turbo charge" their programs. These tips highlight opportunities for making your programs run faster or minimizing the amount of memory that they occupy.

Portability Tip 1.1

We include Portability Tips *to help you write code that will run on a variety of platforms and to explain how Visual C++ achieves its high degree of portability.*

Software Engineering Observation 1.1

The Software Engineering Observations *highlight architectural and design issues that affect the construction of software systems, especially large-scale systems.*

Using Fonts and Colors for Emphasis. We place the key terms and the index's page reference for each defining occurrence in ***bold italic*** text for easier reference. We emphasize on-screen components in the **bold Helvetica** font (e.g., the **File** menu) and emphasize Visual C++ program text in the `Lucida` font (e.g., `int x = 5`).

Web Access. All of the source-code examples for *Visual C++ 2008 How to Program, 2/e* are available for download from:

```
www.deitel.com/books/vcpphtp2/
```

Site registration is quick and easy. Download all the examples, then run each program as you read the corresponding text discussions. Making changes to the examples and seeing the effects of those changes is a great way to enhance your Visual C++ learning experience.

Objectives. Each chapter begins with a statement of objectives. This lets you know what to expect and gives you an opportunity, after reading the chapter, to determine if you have met the objectives.

Quotations. The learning objectives are followed by quotations. Some are humorous; some are philosophical; others offer interesting insights. We hope that you enjoy relating the quotations to the chapter material.

Outline. The chapter outline helps you approach the material in a top-down fashion, so you can anticipate what is to come and set a comfortable and effective learning pace.

Illustrations/Figures. Abundant charts, tables, line drawings, programs and program output are included. We model the flow of control in control statements with UML activity diagrams. UML class diagrams model the fields, constructors and methods of classes. We make extensive use of six major UML diagram types in the optional OOD/UML 2 ATM case study.

Wrap-Up Section. Each of the chapters ends with a brief "wrap-up" section that recaps the chapter content and transitions to the next chapter.

Summary Bullets. Each chapter ends with additional pedagogical features. We present a thorough, bullet-list-style summary of the chapter, section by section.

Terminology. We include an alphabetized list of the important terms defined in each chapter. Each term also appears in the index, with its defining occurrence highlighted with a ***bold, italic*** page number.

Self-Review Exercises and Answers. Extensive self-review exercises and answers are included for self-study.

Exercises. Each chapter concludes with a set of exercises including simple recall of important terminology and concepts; identifying the errors in code samples; writing individual Visual C++ statements; writing small portions of functions and classes; writing complete Visual C++ functions, 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 students and to vary course assignments each semester. Instructors can use these exercises to form homework assignments, short quizzes, major examinations and term projects. See our Programming Projects Resource Center (`www.deitel.com/ProgrammingProjects/`) for many additional exercise and project possibilities.

[***NOTE:*** **Please do not write to us requesting access to the Prentice Hall Instructor's Resource Center, which contains the exercise solutions and the book's ancillaries. Access is limited strictly to college instructors teaching from the book. Instructors may obtain access only through their Prentice Hall representatives.**]

Thousands of Index Entries. We have included an extensive index which is especially useful when you use the book as a reference.

"Double Indexing" of Visual C++ Live-Code Examples. For every source-code program in the book, we index the figure caption both alphabetically and as a subindex item under "Examples." This makes it easier to find examples using particular features.

Object-Oriented Design of an ATM with the UML: A Tour of the Optional Software Engineering Case Study

In this section, we tour the book's optional case study of object-oriented design with the UML. This tour previews the contents of the ten Software Engineering Case Study sections (in Chapters 1, 3–8, 10, 14 and Appendix F). After completing this case study, you'll be thoroughly familiar with a carefully reviewed object-oriented design and implementation for a significant C++ application.

The design presented in the ATM case study was developed at Deitel & Associates, Inc. and scrutinized by a distinguished developmental review team of industry professionals and academics. We crafted this design to meet the requirements of introductory course sequences. Real ATM systems used by banks and their customers worldwide are based on more sophisticated designs that take into consideration many more issues than we have addressed here. Our primary goal throughout the design process was to create a simple design that would be clear to OOD and UML novices, while still demonstrating key OOD concepts and the related UML modeling techniques. We worked hard to keep the design and the code relatively small so that it would work well in the introductory course sequence.

Section 1.21, (Only Required Section of the Case Study) Software Engineering Case Study: Introduction to Object Technology and the UML—introduces the object-oriented design case study with the UML. The section introduces the basic object technology concepts and terminology, including classes, objects, encapsulation, inheritance and polymorphism. We discuss the history of the UML. This is the case study's only required section.

Section 3.8, (Optional) Software Engineering Case Study: Examining the ATM Requirements Specification—discusses a *requirements specification* that specifies the requirements for a system that we'll design and implement—the software for a simple automated teller machine (ATM). We investigate the structure and behavior of object-oriented systems in general. We discuss how the UML will facilitate the design process in subsequent Software Engineering Case Study sections by providing several additional types of diagrams to model our system. We include a list of URLs and book references on object-oriented design with the UML. We discuss the interaction between the ATM system specified by the requirements specification and its user. Specifically, we investigate the scenarios that may occur between the user and the system itself—these are called *use cases*. We model these interactions, using *use case diagrams* of the UML.

Section 4.13, (Optional) Software Engineering Case Study: Identifying the Classes in the ATM Requirements Specification—begins to design the ATM system. We identify its classes, or "building blocks," by extracting the nouns and noun phrases from the requirements specification. We arrange these classes into a UML class diagram that describes the class structure of our simulation. The class diagram also describes relationships, known as *associations*, among classes.

Section 5.13, (Optional) Software Engineering Case Study: Identifying Class Attributes in the ATM System—focuses on the attributes of the classes discussed in Section 4.13. A class contains both *attributes* (data) and *operations* (behaviors). As we'll see in later sections, changes in an object's attributes often affect the object's behavior. To determine the attributes for the classes in our case study, we extract the adjectives describing the nouns and noun phrases (which defined our classes) from the requirements specification, then place the attributes in the class diagram we created in Section 4.13.

Section 6.11, (Optional) Software Engineering Case Study: Identifying Objects' States and Activities in the ATM System—discusses how an object, at any given time, occupies a specific condition called a *state*. A *state transition* occurs when that object receives a message to change state. The UML provides the *state machine diagram*, which identifies the set of possible states that an object may occupy and models that object's state transitions. An object also has an *activity*—the work it performs in its lifetime. The UML provides the *activity diagram*—a flowchart that models an object's activity. In this section, we use both types of diagrams to begin modeling specific behavioral aspects of our ATM system, such as how the ATM carries out a withdrawal transaction and how the ATM responds when the user is authenticated.

Section 7.23, (Optional) Software Engineering Case Study: Identifying Class Operations in the ATM System—identifies the operations, or services, of our classes. We extract from the requirements specification the verbs and verb phrases that specify the operations for each class. We then modify the class diagram of Section 4.13 to include each operation with its associated class. At this point in the case study, we will have gathered all information possible from the requirements specification. However, as future chapters introduce such topics as inheritance, we'll modify our classes and diagrams.

Section 8.15, (Optional) Software Engineering Case Study: Collaboration Among Objects in the ATM System—provides a "rough sketch" of the model for our ATM system. In this section, we see how it works. We investigate the behavior of the simulation by discussing *collaborations*—messages that objects send to each other to communicate. The class operations that we discovered in Section 7.23 turn out to be the collaborations among the objects in our system. We determine the collaborations, then collect them into a *communication diagram*—the UML diagram for modeling collaborations. This diagram reveals which objects collaborate and when. We present a communication diagram of the collaborations among objects to perform an ATM balance inquiry. We then present the

UML *sequence diagram* for modeling interactions in a system. This diagram emphasizes the chronological ordering of messages. A sequence diagram models how objects in the system interact to carry out withdrawal and deposit transactions.

Section 10.12, (Optional) Software Engineering Case Study: Starting to Program the Classes of the ATM System—takes a break from designing the system's behavior. We begin the implementation process to emphasize the material discussed in Chapter 10. Using the UML class diagram of Section 4.13 and the attributes and operations discussed in Section 5.13 and Section 7.23, we show how to implement a class in C++ from a design. We do not implement all classes—because we have not completed the design process. Working from our UML diagrams, we create code for the `Withdrawal` class.

Section 14.11, (Optional) Software Engineering Case Study: Incorporating Inheritance into the ATM System—continues our discussion of object-oriented programming. We consider inheritance—classes sharing common characteristics may inherit attributes and operations from a "base" class. In this section, we investigate how our ATM system can benefit from using inheritance. We document our discoveries in a class diagram that models inheritance relationships—the UML refers to these relationships as *generalizations*. We modify the class diagram of Section 4.13 by using inheritance to group classes with similar characteristics. This section concludes the design of the model portion of our simulation. We fully implement this model in 877 lines of C++ code in Appendix F.

Appendix F, ATM Case Study Code—The majority of the case study involves designing the model (i.e., the data and logic) of the ATM system. In this appendix, we implement that model in C++. Using all the UML diagrams we created, we present the C++ classes necessary to implement the model. We apply the concepts of object-oriented design with the UML and object-oriented programming in C++ that you learned in the chapters. By the end of this appendix, students will have completed the design and implementation of a real-world system, and should feel confident tackling larger systems, such as those that professional software engineers build.

Appendix G, UML 2: Additional Diagram Types—Overviews the UML 2 diagram types that are not found in the OOD/UML Case Study.

Instructor Resources for *Visual C++ 2008 How to Program, 2/e*

Visual C++ 2008 How to Program, 2/e has extensive instructor resources. The Prentice Hall *Instructor's Resource Center* contains the *Solutions Manual* with solutions to the vast majority of the end-of-chapter exercises, a *Test Item File* of multiple-choice questions (approximately two per book section) and PowerPoint® slides containing the code and figures in the text, plus bulleted items that summarize the key points in the text. Instructors can customize the slides. If you are not already a registered faculty member, contact your Prentice Hall representative or visit `vig.prenhall.com/replocator/`.

[*NOTE:* **Please do not write to us requesting access to the Prentice Hall Instructor's Resource Center, which contains the exercise solutions and the book's ancillaries. Access is limited strictly to college instructors teaching from the book. Instructors may obtain access only through their Prentice Hall representatives.**]

Ordering Option for the Education Market

Instructors can order this book packaged with Microsoft Visual C++ 2008 Express for their students. The ISBN for this value pack is 0-13-712940-8.

Deitel® Buzz Online Free E-mail Newsletter

Each week, the *Deitel® Buzz Online* newsletter announces our latest Resource Center(s) and 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. It's also a good way for you to keep posted about issues related to *Visual C++ 2008 How to Program, 2/e*. To subscribe, visit

`www.deitel.com/newsletter/subscribe.html`

The Deitel Online Resource Centers

Our website `www.deitel.com` provides Resource Centers on various topics including programming languages, software, Web 2.0, Internet business and open source projects. (You can see the complete list of Resource Centers in the first few pages of this book.) The Resource Centers evolved out of the research we've done for our books and business endeavors. We've found many exceptional resources online including tutorials, documentation, software downloads, articles, blogs, videos, code samples, books, e-books and more. Most of them are free. In the spirit of Web 2.0, we share these resources with the worldwide community. The Deitel Resource Centers are a starting point for your own research. We help you wade through the vast amount of content on the Internet by providing links to the most valuable resources. Each week we announce our latest Resource Centers in our newsletter, the *Deitel® Buzz Online* (`www.deitel.com/newsletter/subscribe.html`). The following Resource Centers may be of interest to you as you study *Visual C++ 2008 How to Program, 2/e*:

- Visual C++
- Visual Studio Team System
- C++
- C++ Boost Libraries
- C++ Game Programming
- Code Search Engines and Code Sites
- Computer Game Programming
- Computing Jobs
- Open Source
- Programming Projects
- Eclipse
- .NET
- .NET 3.0
- .NET 3.5
- Windows Vista

Acknowledgments

It is a pleasure to acknowledge 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—thanks especially to Abbey Deitel and Barbara Deitel.

We'd also like to thank one of the participants in our Honors Internship program who made significant contributions to this publication—Greg Ayer, a computer science major at Northeastern University. Greg worked on the Collections chapter and the discussion of .NET generics in the Templates and Generics chapter. He also significantly updated the Regular Expressions chapter.

We are fortunate to have worked on this project with the talented and dedicated team of publishing professionals at Prentice Hall. We appreciate the extraordinary efforts of Marcia Horton, Editorial Director of Prentice Hall's Engineering and Computer Science Division. Carole Snyder and Dolores Mars did an extraordinary job recruiting the book's review team and managing the review process. Francesco Santalucia (an independent artist) and Kristine Carney of Prentice Hall did a wonderful job designing the book's cover—we provided the concept, and they made it happen. Bob Engelhardt and Marta Samsel did a marvelous job managing the book's production.

We wish to acknowledge the efforts of our reviewers. Adhering to a tight time schedule, they scrutinized the text and the programs, providing countless suggestions for improving the accuracy and completeness of the presentation.

We sincerely appreciate the efforts of our reviewers:

Visual C++ 2008 How to Program 2/e Reviewers

Microsoft Reviewers: Alvin Chardon, Mykola Dudar, Gordon Hogenson (author of *C++/CLI: The Visual C++ Language for .NET*; Apress), Vytautas Leonavicius and April Reagan. **Academic Reviewers:** Ronald DiNapoli (Cornell University) and Tim H. Lin (California State Polytechnic University, Pomona).

These reviewers scrutinized every aspect of the text and made countless suggestions for improving the accuracy and completeness of the presentation.

This book is heavily based on *C++ How to Program, 6/e*. We'd like to thank the reviewers of the 5th and 6th editions of that book:

C++ How to Program 6/e Reviewers

Academic and Industry Reviewers: Dr. Richard Albright (Goldey-Beacom College), William B. Higdon (University of Indianapolis), Howard Hinnant (Apple), Anne B. Horton (Lockheed Martin), Terrell Hull (Logicalis Integration Solutions), Rex Jaeschke (Independent Consultant), Maria Jump (The University of Texas at Austin), Geoffrey S. Knauth (GNU), Don Kostuch (Independent Consultant), Colin Laplace (Freelance Software Consultant), Stephan T. Lavavej (Microsoft), Amar Raheja (California State Polytechnic University, Pomona), G. Anthony Reina (University of Maryland University College, Europe), Daveed Vandevoorde (C++ Standards Committee), Jeffrey Wiener (DEKA Research & Development Corporation, New Hampshire Community Technical College), and Chad Willwerth (University of Washington, Tacoma). **Ogre Reviewers:** Casey Borders (Sensis Corp.), Gregory Junker (Author of *Pro OGRE3D Programming,* Apress Books), Mark Pope (THQ, Inc.), and Steve Streeting (Torus Knot Software, Ltd.). **Boost/C++OX Reviewers:** Edward Brey (Kohler Co.), Jeff Garland (Boost.org), Douglas Gregor (Indiana University), and Björn Karlsson (Author of *Beyond the C++ Standard Library: An Introduction to Boost*, Addison-Wesley/Readsoft, Inc.).

C++ How to Program 5/e Reviewers

Academic Reviewers: Richard Albright (Goldey-Beacom College), Karen Arlien (Bismarck State College), David Branigan (DeVry University, Illinois), Jimmy Chen (Salt Lake Community College), Martin Dulberg (North Carolina State University), Ric Heishman (Northern Virginia Community College), Richard Holladay (San Diego Mesa College), William Honig (Loyola University), Earl LaBatt (OPNET Technologies, Inc./ University of New Hampshire), Brian Larson (Modesto Junior College), Robert Myers (Florida State University), Gavin Osborne (Saskatchewan Institute of Applied Science and Technology), Wolfgang Pelz (The University of Akron), and Donna Reese (Mississippi State University). **Industry Reviewers:** Curtis Green (Boeing Integrated Defense Systems), Mahesh Hariharan (Microsoft), James Huddleston (Independent Consultant), Ed James-Beckham (Borland Software Corporation), Don Kostuch (Independent Consultant), Meng Lee (Hewlett-Packard), Kriang Lerdsuwanakij (Siemens Limited), William Mike Miller (Edison Design Group, Inc.), Mark Schimmel (Borland International), Vicki Scott (Metrowerks), James Snell (Boeing Integrated Defense Systems), and Raymond Stephenson (Microsoft). **OOD/UML Optional Software Engineering Case Study Reviewers:** Sinan Si Alhir (Independent Consultant), Karen Arlien (Bismarck State College), David Branigan (DeVry University, Illinois), Martin Dulberg (North Carolina State University), Ric Heishman (Northern Virginia Community College), Richard Holladay (San Diego Mesa College), Earl LaBatt (OPNET Technologies, Inc./University of New Hampshire), Brian Larson (Modesto Junior College), Gavin Osborne (Saskatchewan Institute of Applied Science and Technology), Praveen Sadhu (Infodat International, Inc.), Cameron Skinner (Embarcadero Technologies, Inc./OMG), and Steve Tockey (Construx Software).

Well, there you have it! Welcome to the exciting world of Visual C++ and object-oriented programming. We hope you enjoy this look at contemporary computer programming. Good luck!

As you read the book, we would sincerely appreciate your comments, criticisms, corrections and suggestions for improving the text. Please address all correspondence to:

```
deitel@deitel.com
```

We'll respond promptly, and post corrections and clarifications on:

```
www.deitel.com/books/vcpphtp2/
```

We hope you enjoy reading *Visual C++ 2008 How to Program, Second Edition* as much as we enjoyed writing it!

Paul J. Deitel
Dr. Harvey M. Deitel
Dan T. Quirk
Maynard, Massachusetts
December 2007

About the Authors

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT's Sloan School of Management, where he studied Information Technology. He holds the Java Certified Programmer and Java Certified Developer certifications, and has been designated by Sun Microsystems as a Java Champion. Through Deitel & Associates, Inc., he has delivered C++, C#, VB, C and Java courses to industry clients, including Cisco, IBM, Sun Microsystems, Dell, Lucent Technologies, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, Stratus, Cambridge Technology Partners, Open Environment Corporation, One Wave, Hyperion Software, Adra Systems, Entergy, CableData Systems, Nortel Networks, Puma, iRobot, Invensys and many more. He has also lectured on Java and C++ for the Boston Chapter of the Association for Computing Machinery. He and his father, Dr. Harvey M. Deitel, are the world's best-selling programming language textbook authors.

Dr. Harvey M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 46 years of academic and industry experience in the computer field. Dr. Deitel earned B.S. and M.S. degrees from MIT and a Ph.D. from Boston University. He has 20 years of college teaching experience, including earning tenure and serving as the Chairman of the Computer Science Department at Boston College before founding Deitel & Associates, Inc., with his son, Paul J. Deitel. He and Paul are the co-authors of several dozen books and multimedia packages and they are writing many more. The Deitels' texts have earned international recognition with translations published in Japanese, German, Russian, Spanish, Traditional Chinese, Simplified Chinese, Korean, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of professional seminars to major corporations, academic institutions, government organizations and the military.

Dan T. Quirk is a senior at the University of Rochester completing a B.S. in Computer Science. His industry experience includes web application development, database development and networking. His course work includes artificial intelligence, systems programming, computational theory, operating systems and various programming languages.

About Deitel & Associates, Inc.

Deitel & Associates, Inc., is an internationally recognized corporate training and content-creation organization specializing in computer programming languages, Internet and World Wide Web software technology, object technology education and Internet business development through its Web 2.0 Internet Business Initiative. The company provides instructor-led courses on major programming languages and platforms, such as C++, Java, C, C#, Visual C++, Visual Basic, XML, Perl, object technology and Internet and World Wide Web programming. 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 companies, government agencies, branches of the military, and academic institutions. Through its 31-year publishing partnership with Prentice Hall, Deitel & Associates, Inc. publishes leading-edge programming textbooks, professional books, interactive multimedia *Cyber Classrooms*, *LiveLessons* video courses, web-based training courses and e-content for the popular course management systems WebCT, Blackboard and Pearson's Course-Compass. 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 *Dive Into*® Series Corporate Training curriculum, visit:

```
www.deitel.com
```

and subscribe to the free *Deitel*® *Buzz Online* e-mail newsletter at:

```
www.deitel.com/newsletter/subscribe.html
```

Check out the growing list of online Deitel Resource Centers at:

```
www.deitel.com/resourcecenters.html
```

Individuals wishing to purchase Deitel publications can do so through:

```
www.deitel.com
```

Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Prentice Hall. For more information, visit

```
www.prenhall.com/mischtm/support.html#order
```