Preface

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

For the Student

Welcome to the C++ computer programming language and *C++ How to Program, Eighth Edition*! This book presents leading-edge computing technologies, and is particularly appropriate for inroductory course sequences based on the curriculum recommendations of two key professional organizations—the ACM and the IEEE.

The new Chapter 1 presents intriguing facts and figures. Our goal is to get you excited about studying computers and programming. The chapter includes a table of some of the research made possible by computers; current technology trends and hardware discussions; the data hierarchy; social networking; a table of business and technology publications and websites that will help you stay up-to-date with the latest technology news, trends and career opportunities; additional Making a Difference exercises and more.

We focus on software engineering best practices. At the heart of the book is our signature "live-code approach"—programming concepts are presented in the context of complete working programs, rather than in code snippets. Each C++ code example is accompanied by live sample executions, so you can see exactly what each program does when it's run on a computer. All the source code is available at `www.deitel.com/books/cpphtp8/` and `www.pearsonhighered.com/deitel/`.

Much of this Preface is addressed to instructors. Please be sure to read the sections entitled Pedagogic Features; Teaching Approach; Software Used in *C++ How to Program, 8/e*; C++ IDE Resource Kit and CourseSmart Web Books.

We believe that this book and its support materials will give you an informative, interesting, challenging and entertaining C++ educational 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, visit `www.deitel.com/books/cpphtp8/`, follow us on Facebook (`www.deitel.com/deitelfan`) and Twitter (`@deitel`), and subscribe to the *Deitel® Buzz Online* newsletter (`www.deitel.com/newsletter/subscribe.html`). Good luck!

New and Updated Features

Here are the updates we've made for *C++ How to Program, 8/e*:

Impending New C++ Standard

- ***Optional sections.*** We cover various features of the new standard (sometimes called C++0x and due late in 2011 or early in 2012) in *optional modular sections* and in Chapter 23. These are *easy to include or omit*. Popular compilers such as Microsoft Visual C++ 2010 and GNU C++ 4.5 already implement many of these features. To enable the new standard features in GNU C++, use the `-std=C++0x` flag when you compile the corresponding programs.

- ***Boost C++ Libraries, Technical Report 1 (TR1) and C++0x.*** In Chapter 23, we introduce the Boost C++ Libraries, Technical Report 1 (TR1) and C++0x. The free Boost open source libraries are created by members of the C++ community. Technical Report 1 describes the proposed changes to the C++ Standard Library, many of which are based on current Boost libraries. The C++ Standards Committee is revising the C++ Standard—the main goals are to make C++ easier to learn, improve library building capabilities, and increase compatibility with the C programming language. The new standard will include many of the libraries in TR1 and changes to the core language. We overview the Boost libraries and provide code examples for the "regular expression" and "smart pointer" libraries. Regular expressions are used to match specific character patterns in text. They can be used, for example, to validate data to ensure that it's in a particular format, to replace parts of one string with another, or to split a string. Many common bugs in C and C++ code are related to pointers, a powerful programming capability you'll study in Chapter 8. Smart pointers help you avoid errors by providing additional functionality to standard pointers.
- ***`unique_ptr` vs. `auto_ptr`.*** We replaced our `auto_ptr` example with the impending standard's class `unique_ptr`, which fixes various problems that were associated with class `auto_ptr`. Use of `auto_ptr` is deprecated and `unique_ptr` is already implemented in many popular compilers, including Visual C++ 2010 and GNU C++ 4.5.
- ***Initializer lists for user-defined types.*** These enable objects of your own types to be initialized using the same syntax as built-in arrays.
- ***Range-based `for` statement.*** A version of the `for` statement that iterates over all the elements of an array or container (such as an object of the `vector` class).
- ***Lambda expressions.*** These enable you to create anonymous functions that can be passed to other functions as arguments.
- ***`auto` storage class specifier.*** The keyword `auto` can no longer be used as a storage class specifier.
- ***`auto`.*** This keyword now deduces the type of a variable from its initializer.
- ***`nullptr`.*** This keyword is a replacement for assigning zero to a null pointer.
- ***`static_assert`.*** This capability allows you to test certain aspects of the program at compile time.
- ***New `long long` and `unsigned long long` types.*** These new types were introduced for use with 64-bit machines.

Pedagogic Features

- ***Enhanced Making a Difference exercises set.*** We encourage you to use computers and the Internet to research and solve significant social problems. These exercises are meant to increase awareness and discussion of important issues the world is facing. We hope you'll approach them with your own values, politics and beliefs.

Check out our new Making a Difference Resource Center at `www.deitel.com/MakingADifference` for additional ideas you may want to investigate further.

- ***Page numbers for key terms in chapter summaries.*** For key terms that appear in the chapter summaries, we include the page number of each term's defining occurrence in the chapter.
- ***VideoNotes.*** The Companion Website includes 15+ hours of VideoNotes in which co-author Paul Deitel explains in detail most of the programs in the core chapters. Instructors have told us that their students find the VideoNotes valuable for preparing for and reviewing lectures.
- ***Modular presentation.*** We've grouped the chapters into teaching modules. The Chapter Dependency Chart (later in this Preface) reflects the modularization.

Object Technology

- ***Object-oriented programming and design.*** We introduce the basic concepts and terminology of object technology in Chapter 1. Students develop their first customized classes and objects in Chapter 3. Presenting objects and classes early gets students "thinking about objects" immediately and mastering these concepts more thoroughly. [For courses that require a late-objects approach, consider *C++ How to Program, Late Objects Version, Seventh Edition*, which begins with six chapters on programming fundamentals (including two on control statements) and continues with seven chapters that gradually introduce object-oriented programming concepts.]
- ***Integrated case studies.*** We provide several case studies that span multiple sections and chapters. These include development of the `GradeBook` class in Chapters 3–7, the `Time` class in Chapters 9–10, the `Employee` class in Chapters 12–13, and the optional OOD/UML ATM case study in Chapters 25–26.
- ***Integrated `GradeBook` case study.*** The `GradeBook` case study uses classes and objects in Chapters 3–7 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.
- ***Exception handling.*** We integrate basic exception handling early in the book. Instructors can easily pull more detailed material forward from Chapter 16, Exception Handling: A Deeper Look.
- ***Prefer `vector`s to C arrays.*** C++ offers two types of arrays—`vector` class objects (which we start using in Chapter 7) and C-style, pointer-based arrays. As appropriate, we use class template `vector` instead of C arrays throughout the book. However, we begin by discussing C arrays in Chapter 7 to prepare you for working with legacy code and to use as a basis for building your own customized `Array` class in Chapter 11.
- ***Prefer `string` objects to C strings.*** Similarly, C++ offers two types of strings—`string` class objects (which we use starting in Chapter 3) and C-style, pointer-based strings. We continue to include some early discussions of C strings to give

you practice with pointer manipulations, to illustrate dynamic memory allocation with `new` and `delete` and to prepare you for working with C strings in the legacy code that you'll encounter in industry. In new development, you should favor `string` class objects. We've replaced most occurrences of C strings with instances of C++ class `string` to make programs more robust and eliminate many of the security problems that can be caused by using C strings.

- ***Optional case study: Using the UML to develop an object-oriented design and C++ implementation of an ATM.*** The UML™ (Unified Modeling Language™) is the industry-standard graphical language for modeling object-oriented systems. Chapters 25–26 include an *optional* online case study on object-oriented design using the UML. We design and implement the software for a simple automated teller machine (ATM). We analyze a typical requirements document that specifies the system to be built. We determine the classes needed to implement that system, the attributes the classes need to have, the behaviors the classes need to exhibit and specify how the classes must interact with one another to meet the system requirements. From the design we produce a complete C++ implementation. Students often report having a "light-bulb moment"—the case study helps them "tie it all together" and really understand object orientation.
- ***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 22 introduces the STL and discusses its three key components—containers, iterators and algorithms. The STL components provide tremendous expressive power, often reducing many lines of code to a single statement.

Other Features

- ***Printed book contains core content; additional chapters are online.*** Several online chapters are included for more advanced courses and for professionals. These are available in searchable PDF format on the book's password-protected Companion Website—see the access card in the front of this book.
- ***Reorganized Chapter 11, Operator Overloading; Class `string`.*** We reorganized this chapter to begin with standard library class `string` so readers can see an elegant use of operator overloading before they implement their own. We also moved the section on proxy classes to the end of Chapter 10, where it's a more natural fit.
- ***Enhanced use of `const`.*** We increased the use of `const` book-wide to encourage better software engineering.
- ***Software engineering concepts.*** Chapter 1 briefly introduces very current software engineering terminology, including agile software development, Web 2.0, Ajax, SaaS (Software as a Service), PaaS (Platform as a Service), cloud computing, web services, open source software, design patterns, refactoring, LAMP and more.
- ***Compilation and linking process for multiple-source-file programs.*** Chapter 3 includes a detailed diagram and discussion of the compilation and linking process that produces an executable program.

- ***Function Call Stack Explanation.*** In Chapter 6, we provide a detailed discussion with illustrations of the function call stack and activation records to explain how 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.
- ***Tuned Treatment of Inheritance and Polymorphism.*** Chapters 12–13 have been carefully tuned using a concise `Employee` class hierarchy. We use this same treatment in our C++, Java, C# and Visual Basic books—one of our reviewers called it the best he had seen in 25 years as a trainer and consultant.
- ***Discussion and illustration of how polymorphism works "under the hood."*** Chapter 13 contains a detailed diagram and explanation of how C++ can implement polymorphism, `virtual` functions and dynamic binding internally. This gives students a solid understanding of how these capabilities work.
- ***ISO/IEC C++ standard compliance.*** We've audited our presentation against the ISO/IEC C++ standard document.
- ***Debugger appendices.*** We provide two Using the Debugger appendices on the book's Companion Website—Appendix H, Using the Visual Studio Debugger, and Appendix I, Using the GNU C++ Debugger.
- ***Code tested on multiple platforms.*** We tested the code examples on various popular C++ platforms including GNU C++ on Linux and Microsoft Windows, and Visual C++ on Windows. For the most part, the book's examples port to popular standard-compliant compilers.
- ***Game Programming.*** Because of limited interest, we've removed from the book Chapter 27, Game Programming with Ogre (which covers only Linux). For instructors who would like to continue using this material with *C++ How to Program, 8/e*, we've included the version from *C++ How to Program, 7/e* on the book's Companion Website.

Our Text + Digital Approach to Content

We surveyed hundreds of instructors teaching C++ courses and learned that most want a book with content focused on their introductory courses. With that in mind, we moved various advanced chapters to the web. Having this content in digital format makes it easily searchable, and gives us the ability to fix errata and add new content as appropriate. The book's Companion Website, which is accessible at

```
www.pearsonhighered.com/deitel/
```

(see the access card at the front of the book) contains the following chapters in *searchable* PDF format:

- Chapter 25, ATM Case Study, Part 1: Object-Oriented Design with the UML
- Chapter 26, ATM Case Study, Part 2: Implementing an Object-Oriented Design
- Game Programming with Ogre (from *C++ How to Program, 7/e*)
- Appendix F, C Legacy Code Topics

- Appendix G, UML 2: Additional Diagram Types
- Appendix H, Using the Visual Studio Debugger
- Appendix I, Using the GNU C++ Debugger

The Companion Website also includes:

- Extensive VideoNotes—watch and listen as co-author Paul Deitel discusses the key features of the code examples in Chapters 2–13 and portions of Chapters 16 and 17.
- Two true/false questions per section with answers for self-review.
- Solutions to approximately half of the solved exercises in the book.

The following materials are posted at the Companion Website and at `www.deitel.com/books/cpphtp8/`:

- An array of function pointers example and additional function pointer exercises (from Chapter 8).
- `String` Class Operator Overloading Case Study (from Chapter 11).
- Building Your Own Compiler exercise descriptions (from Chapter 20).

Dependency Chart

The chart on the next page shows the dependencies among the chapters to help instructors plan their syllabi. *C++ How to Program, 8/e* is appropriate for CS1 and CS2 courses.

Teaching Approach

C++ How to Program, 8/e, contains a rich collection of examples. We stress program clarity and concentrate on building well-engineered software.

Live-code approach. The book is loaded with "live-code" examples—most new concepts are presented in the context of *complete working C++ applications*, followed by one or more executions showing program inputs and outputs. In the few cases where we use a code snippet, we tested it in a complete working program, then copied and pasted it into the book.

Syntax coloring. For readability, we syntax color all the C++ code, similar to the way most C++ integrated-development environments and code editors syntax color code. Our coloring conventions are as follows:

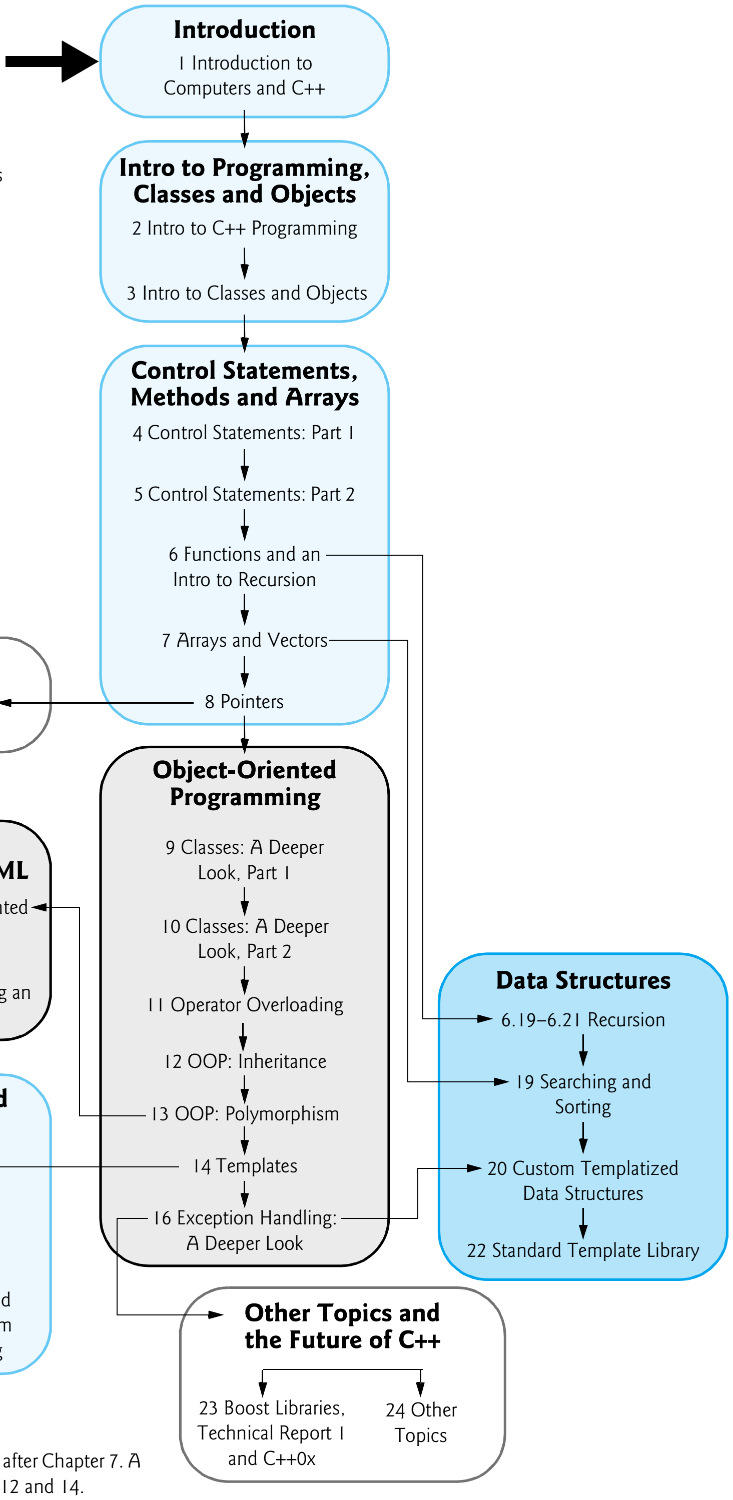
```
comments appear like this
keywords appear like this
constants and literal values appear like this
all other code appears in black
```

Code highlighting. We place light blue shaded rectangles around each program's key code segments.

Using fonts for emphasis. We place the key terms and the index's page reference for each defining occurrence in **bold blue** text for easy reference. We emphasize on-screen components in the **bold Helvetica** font (e.g., the **File** menu) and C++ program text in the `Lucida` font (for example, `int x = 5;`).

Chapter Dependency Chart

[*Note:* Arrows pointing into a chapter indicate that chapter's dependencies.]



Introduction
1 Introduction to Computers and C++
Intro to Programming, Classes and Objects
2 Intro to C++ Programming
3 Intro to Classes and Objects
Control Statements, Methods and Arrays
4 Control Statements: Part 1
5 Control Statements: Part 2
6 Functions and an Intro to Recursion
7 Arrays and Vectors
8 Pointers
Legacy C Topics
21 Bits, Characters, C-Strings and structs
Object-Oriented Programming
9 Classes: A Deeper Look, Part 1
10 Classes: A Deeper Look, Part 2
11 Operator Overloading
12 OOP: Inheritance
13 OOP: Polymorphism
14 Templates
16 Exception Handling: A Deeper Look
Object-Oriented Design with the UML
25 (Optional) Object-Oriented Design with the UML
26 (Optional) Implementing an Object-Oriented Design
Streams, Files and Strings
15 Stream Input/Output[1]
17 File Processing
18 Class string and String Stream Processing
Data Structures
6.19–6.21 Recursion
19 Searching and Sorting
20 Custom Templatized Data Structures
22 Standard Template Library
Other Topics and the Future of C++
23 Boost Libraries, Technical Report 1 and C++0x
24 Other Topics

1. Most of Chapter 15 is readable after Chapter 7. A small portion requires Chapters 12 and 14.

Objectives. The opening quotes are followed by a list of chapter objectives.

Illustrations/ figures. Abundant tables, line drawings, UML diagrams, programs and program outputs are included.

Programming tips. We include programming tips to help you focus on important aspects of program development. These tips and practices represent the best we've gleaned from a combined seven decades of programming and teaching experience.

Good Programming Practices

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

Common Programming Errors

Pointing out these Common Programming Errors *reduces the likelihood that you'll make them.*

Error-Prevention Tips

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

Performance Tips

These tips highlight opportunities for making your programs run faster or minimizing the amount of memory that they occupy.

Portability Tips

The Portability Tips *help you write code that will run on a variety of platforms.*

Software Engineering Observations

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

Summary bullets. We present a section-by-section bullet-list summary of the chapter with the page references to the defining occurrence for many of the key terms in each section.

Self-review exercises and answers. Extensive self-review exercises *and* answers are included for self study. All of the exercises in the optional ATM case study are fully solved.

Exercises. Each chapter concludes with a substantial set of exercises including:

- simple recall of important terminology and concepts
- What's wrong with this code?
- What does this code do?
- writing individual statements and small portions of functions and classes
- writing complete functions, classes and programs
- major projects.

Please do not write to us requesting access to the Pearson Instructor's Resource Center which contains the book's instructor supplements, including the exercise solutions. Ac-

cess is limited strictly to college instructors teaching from the book. Instructors may obtain access only through their Pearson representatives. Solutions are *not* provided for "project" exercises. Check out our Programming Projects Resource Center for lots of additional exercise and project possibilities (`www.deitel.com/ProgrammingProjects/`).

Index. We've included an extensive index. Defining occurrences of key terms are highlighted with a **bold blue** page number.

Software Used in C++ *How to Program, 8/e*

We wrote *C++ How to Program, 8/e* using Microsoft's free Visual C++ Express Edition (which is available free for download at `www.microsoft.com/express/downloads/`) and the free GNU C++ (`gcc.gnu.org/install/binaries.html`), which is already installed on most Linux systems and can be installed on Mac OS X and Windows systems. Apple includes GNU C++ in their Xcode development tools, which Mac OS X users can download from `developer.apple.com/technologies/tools/xcode.html`.

C++ IDE Resource Kit

Your instructor may have ordered through your college bookstore a Value Pack edition of *C++ How to Program, 8/e* that comes bundled with the C++ IDE Resource Kit. This kit contains CD or DVD versions of:

- Microsoft® Visual Studio 2010 Express Edition (`www.microsoft.com/express/`)
- Dev C++ (`www.bloodshed.net/download.html`)
- NetBeans (`netbeans.org/downloads/index.html`)
- Eclipse (`eclipse.org/downloads/`)
- CodeLite (`codelite.org/LiteEditor/Download`)

You can download these software packages from the websites specified above. The C++ IDE Resource Kit also includes access to a Companion Website containing step-by-step written instructions and VideoNotes to help you get started with each development environment. If your book did not come with the C++ IDE Resource Kit, you can purchase access to the Resource Kit's Companion Website from `www.pearsonhighered.com/cppidekit/`.

CourseSmart Web Books

Today's students and instructors have increasing demands on their time and money. Pearson has responded to that need by offering digital texts and course materials online through CourseSmart. CourseSmart allows faculty to review course materials online, saving time and costs. It offers students a high-quality digital version of the text for less than the cost of a print copy of the text. Students receive the same content offered in the print textbook enhanced by search, note-taking, and printing tools. For more information, visit `www.coursesmart.com`.

Instructor Supplements

The following supplements are available to qualified instructors only through Pearson Education's Instructor Resource Center (`www.pearsonhighered.com/irc`):

- *Solutions Manual* with solutions to the vast majority of the end-of-chapter exercises and Lab Manual exercises. We've added dozens of Making a Difference exercises, most with solutions.
- *Test Item File* of multiple-choice questions (approximately two per book section)
- Customizable PowerPoint® slides containing all the code and figures in the text, plus bulleted items that summarize the key points in the text

If you're not already a registered faculty member, contact your Pearson representative or visit `www.pearsonhighered.com/educator/replocator/`.

Acknowledgments2

We'd like to thank Abbey Deitel and Barbara Deitel of Deitel & Associates, Inc. for long hours devoted to this project. We're fortunate to have worked with the dedicated team of publishing professionals at Pearson. We appreciate the guidance, savvy and energy of Michael Hirsch, Editor-in-Chief of Computer Science. Carole Snyder recruited the book's reviewers and managed the review process. Bob Engelhardt managed the book's production.

Reviewers
We wish to acknowledge the efforts of our seventh and eighth edition reviewers. They scrutinized the text and the programs and provided countless suggestions for improving the presentation: Virginia Bailey (Jackson StateUniversity), Thomas J. Borrelli (Rochester Institute of Technology), Chris Cox (Adobe Systems), Gregory Dai (eBay), Peter J. DePasquale (The College of New Jersey), John Dibling (SpryWare), Susan Gauch (University of Arkansas), Doug Gregor (Apple, Inc.), Jack Hagemeister (Washington State University), Williams M. Higdon (University of Indiana), Wing-Ning Li (University of Arkansas), Dean Mathias (Utah State University), Robert A. McLain (Tidewater Community College), April Reagan (Microsoft), José Antonio González Seco (Parliament of Andalusia, Spain), Dave Topham (Ohlone College) and Anthony Williams (author and C++ Standards Committee member).

Well, there you have it! 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. We hope you enjoy working with *C++ How to Program, Eighth Edition* as much as we enjoyed writing it!

Paul and Harvey Deitel

About the Authors

Paul J. Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., is a graduate of MIT, where he studied Information Technology. Through Deitel & Associates, Inc., he has delivered hundreds of C++, Java, C#, Visual Basic, C and Internet programming courses to industry clients, including Cisco, IBM, Siemens, 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, SunGard Higher Education, Stratus, Cambridge Technology Partners, One Wave, Hyperion Software, Adra Systems, Entergy, CableData Systems, Nortel Networks, Puma, iRobot, Invensys and many more. He and his co-author, 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 50 years of experience in the computer field. Dr. Deitel earned B.S. and M.S. degrees from MIT in Electrical Engineering and a Ph.D. in Mathematics from Boston University—at both he studied computing before separate computer science degree programs were created. He has extensive 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 dozens of books and LiveLessons multimedia packages. With translations published in Japanese, German, Russian, Chinese, Spanish, Korean, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish, the Deitels' texts have earned international recognition. Dr. Deitel has delivered hundreds of professional programming language seminars to major corporations, academic institutions, government organizations and the military.

About Deitel & Associates, Inc.

Deitel & Associates, Inc., is an internationally recognized corporate training and authoring organization specializing in computer programming languages, Internet and web software technology, object-technology and Android™ and iPhone® education and applications development. The company provides instructor-led courses delivered at client sites worldwide on major programming languages and platforms, such as C++, Visual C++®, C, Java™, Visual C#®, Visual Basic®, XML®, Python®, object technology, Internet and web programming, Android and iPhone app development, and a growing list of additional programming and software-development courses. The founders of Deitel & Associates, Inc., are Paul J. Deitel and Dr. Harvey M. Deitel. The company's clients include many of the world's largest corporations, government agencies, branches of the military, and academic institutions. Through its 35-year publishing partnership with Prentice Hall/Pearson Higher Education, Deitel & Associates, Inc., publishes leading-edge programming textbooks, professional books, interactive multimedia *Cyber Classrooms*, and *LiveLessons* DVD-based and web-based video courses. 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 *Dive Into® Series* Corporate Training curriculum delivered at client locations worldwide, visit:

```
www.deitel.com/training/
```

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

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

and follow the authors on Facebook (`www.deitel.com/deitelfan`) and Twitter (`@deitel`).

Individuals wishing to purchase Deitel books, and *LiveLessons* DVD and web-based training courses can do so through `www.deitel.com`. Bulk orders by corporations, the government, the military and academic institutions should be placed directly with Pearson. For more information, visit

```
    www.pearsonhighered.com
```