“The chief merit of language is clearness …”
—Galen

Welcome to the C++ computer programming language and C++ How to Program, Ninth Edition. This book presents leading-edge computing technologies. It’s appropriate for introductory course sequences based on the curriculum recommendations of two key professional organizations—the ACM and the IEEE. If you haven’t already done so, please read the back cover and inside back cover—these capture the essence of the book concisely. In this Preface we provide more detail for students, instructors and professionals.

At the heart of the book is the Deitel signature live-code approach—we present concepts in the context of complete working programs followed by sample executions, rather than in code snippets. Read the online Before You Begin section (www.deitel.com/books/cpphtp9/cpphtp9_BYB.pdf) to learn how to set up your Linux-based, Windows-based or Apple OS X-based computer to run the hundreds of code examples. All the source code is available at www.deitel.com/books/cpphtp9 and www.pearsonhighered.com/deitel. Use the source code we provide to run each program as you study it.

We believe that this book and its support materials will give you an informative, challenging and entertaining introduction to C++. As you read the book, if you have questions, we’re easy to reach at deitel@deitel.com—we’ll respond promptly. For book updates, visit www.deitel.com/books/cpphtp9, join our social media communities on Facebook (www.deitel.com/DeitelFan), Twitter (@deitel), Google+ (gplus.to/deitel) and LinkedIn (bit.ly/DeitelLinkedIn), and subscribe to the Deitel® Buzz Online newsletter (www.deitel.com/newsletter/subscribe.html).

**C++11 Standard**

The new C++11 standard, published in 2011, motivated us to write C++ How to Program, 9/e. Throughout the book, each new C++11 feature is marked with the “11” icon you see here in the margin. These are some of the key C++11 features of this new edition:

- **Conforms to the new C++11 standard.** Extensive coverage of the new C++11 features (Fig. 1).

- **Code thoroughly tested on three popular industrial-strength C++11 compilers.** We tested the code examples on GNU™ C++ 4.7, Microsoft® Visual C++® 2012 and Apple® LLVM in Xcode® 4.5.

- **Smart pointers.** Smart pointers help you avoid dynamic memory management errors by providing additional functionality beyond that of built-in pointers. We discuss unique_ptr in Chapter 17, and shared_ptr and weak_ptr in Chapter 24.
Earlier coverage of Standard Library containers, iterators and algorithms, enhanced with C++11 capabilities. We moved the treatment of Standard Library containers, iterators and algorithms from Chapter 22 in the previous edition to Chapters 15 and 16 and enhanced it with additional C++11 features. The vast majority of your data structure needs can be fulfilled by reusing these Standard Library capabilities. We’ll show you how to build your own custom data structures in Chapter 19.

Online Chapter 24, C++11: Additional Topics. In this chapter, we present additional C++11 topics. The new C++11 standard has been available since 2011, but not all C++ compilers have fully implemented the features. If all three of our key compilers already implemented a particular C++11 feature at the time we wrote this book, we generally integrated a discussion of that feature into the text with a live-code example. If any of these compilers had not implemented that feature, we included a bold italic heading followed by a brief discussion of the feature. Many of those discussions are expanded in online Chapter 24 as the features are imple-
mented. This chapter includes discussions of regular expressions, shared_ptr and weak_ptr smart pointers, move semantics and more.

- **Random Number generation, simulation and game playing.** To help make programs more secure, we've added a treatment of C++11’s new non-deterministic random-number generation capabilities.

**Object-Oriented Programming**

- **Early-objects approach.** The book introduces the basic concepts and terminology of object technology in Chapter 1. You'll develop your first customized classes and objects in Chapter 3. Presenting objects and classes early gets you “thinking about objects” immediately and mastering these concepts more thoroughly.¹

- **C++ Standard Library string.** C++ offers two types of strings—string class objects (which we begin using in Chapter 3) and C strings. 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 of C strings. We continue to discuss C strings later in the book to prepare you for working with the legacy code that you'll encounter in industry. In new development, you should favor string objects.

- **C++ Standard Library array.** Our primary treatment of arrays now uses the Standard Library’s array class template instead of built-in, C-style, pointer-based arrays. We still cover built-in arrays because they remain useful in C++ and so that you'll be able to read legacy code. C++ offers three types of arrays—arrays and vectors (which we start using in Chapter 7) and C-style, pointer-based arrays which we discuss in Chapter 8. As appropriate, we use class template array instead of C arrays throughout the book. In new development, you should favor class template array objects.

- **Crafting valuable classes.** A key goal of this book is to prepare you to build valuable classes. In the Chapter 10 case study, you'll build your own custom Array class, then in the Chapter 18 exercises you'll convert it to a class template. You'll truly appreciate the class concept. Chapter 10 begins with a test-drive of class template string so you can see an elegant use of operator overloading before you implement your own customized class with overloaded operators.

- **Case studies in object-oriented programming.** We provide case studies that span multiple sections and chapters and cover the software development lifecycle. These include the GradeBook class in Chapters 3–7, the Time class in Chapter 9 and the Employee class in Chapters 11–12. Chapter 12 contains a detailed diagram and explanation of how C++ can implement polymorphism, virtual functions and dynamic binding “under the hood.”

- **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

¹ For courses that require a late-objects approach, consider C++ How to Program, Late Objects Version, 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.
industry-standard graphical language for modeling object-oriented systems. We introduce the UML in the early chapters. Online Chapters 25 and 26 include an optional 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 we 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 that the case study helps them “tie it all together” and truly understand object orientation.

- **Exception handling.** We integrate basic exception handling early in the book. Instructors can easily pull more detailed material forward from Chapter 17, Exception Handling: A Deeper Look.
- **Custom template-based data structures.** We provide a rich multi-chapter treatment of data structures—see the Data Structures module in the chapter dependency chart (Fig. 6).
- **Three programming paradigms.** We discuss structured programming, object-oriented programming and generic programming.

**Pedagogic Features**

- **Rich coverage of C++ fundamentals.** We include a clear two-chapter treatment of control statements and algorithm development.
- **Chapter 2 provides a simple introduction to C++ programming.**
- **Examples.** We include a broad range of example programs selected from computer science, business, simulation, game playing and other topics (Fig. 2).

### Examples

<table>
<thead>
<tr>
<th>Array class case study</th>
<th>Craps dice game simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author class</td>
<td>Credit inquiry program</td>
</tr>
<tr>
<td>Bank account program</td>
<td>Date class</td>
</tr>
<tr>
<td>Bar chart printing program</td>
<td>Downcasting and runtime type information</td>
</tr>
<tr>
<td>BasePlusCommissionEmployee class</td>
<td>Employee class</td>
</tr>
<tr>
<td>Binary tree creation and traversal</td>
<td>explicit constructor</td>
</tr>
<tr>
<td>BinarySearch test program</td>
<td>fibonacci function</td>
</tr>
<tr>
<td>Card shuffling and dealing</td>
<td>fill algorithms</td>
</tr>
<tr>
<td>ClientData class</td>
<td>Function-template specializations of function</td>
</tr>
<tr>
<td>CommissionEmployee class</td>
<td>template printArray</td>
</tr>
<tr>
<td>Comparing strings</td>
<td>generate algorithms</td>
</tr>
<tr>
<td>Compilation and linking process</td>
<td>GradeBook Class</td>
</tr>
<tr>
<td>Compound interest calculations with for</td>
<td>Initializing an array in a declaration</td>
</tr>
<tr>
<td>Converting string objects to C strings</td>
<td>Input from an istringstream object</td>
</tr>
<tr>
<td>Counter-controlled repetition</td>
<td>Iterative factorial solution</td>
</tr>
</tbody>
</table>

*Fig. 2* | A sampling of the book’s examples. (Part 1 of 2.)
Audience. The examples are accessible to computer science, information technology, software engineering and business students in novice-level and intermediate-level C++ courses. The book is also used by professional programmers.

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

Interesting, entertaining and challenging exercises. Each chapter concludes with a substantial set of exercises, including simple recall of important terminology and concepts, identifying the errors in code samples, writing individual program statements, writing small portions of C++ classes and member and non-member functions, writing complete programs and implementing major projects. Figure 3 lists a sampling of the book’s exercises, including our Making a Difference exercises, which encourage you to use computers and the Internet to research and solve significant social problems. We hope you’ll approach these exercises with your own values, politics and beliefs.

<table>
<thead>
<tr>
<th>Examples</th>
<th>Exercises</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lambda expressions</td>
<td>SalesPerson class</td>
</tr>
<tr>
<td>Linked list manipulation</td>
<td>Searching and sorting algorithms of the Standard Library</td>
</tr>
<tr>
<td>map class template</td>
<td>Sequential files</td>
</tr>
<tr>
<td>Mathematical algorithms of the Standard Library</td>
<td>set class template</td>
</tr>
<tr>
<td>maximum function template</td>
<td>shared_ptr program</td>
</tr>
<tr>
<td>Merge sort program</td>
<td>stack adapter class</td>
</tr>
<tr>
<td>multiset class template</td>
<td>Stack class</td>
</tr>
<tr>
<td>new throwing bad_alloc on failure</td>
<td>Stack unwinding</td>
</tr>
<tr>
<td>PhoneNumber class</td>
<td>Standard Library string class program</td>
</tr>
<tr>
<td>Poll analysis program</td>
<td>Stream manipulator showbase</td>
</tr>
<tr>
<td>Polymorphism demonstration</td>
<td>string assignment and concatenation</td>
</tr>
<tr>
<td>Preincrementing and postincrementing</td>
<td>string member function substr</td>
</tr>
<tr>
<td>priority_queue adapter class</td>
<td>Summing integers with the for statement</td>
</tr>
<tr>
<td>queue adapter class</td>
<td>Time class</td>
</tr>
<tr>
<td>Random-access files</td>
<td>unique_ptr object managing dynamically allocated memory</td>
</tr>
<tr>
<td>Random number generation</td>
<td>Validating user input with regular expressions</td>
</tr>
<tr>
<td>Recursive function factorial</td>
<td>vector class template</td>
</tr>
<tr>
<td>Rolling a six-sided die 6,000,000 times</td>
<td></td>
</tr>
<tr>
<td>SalariedEmployee class</td>
<td></td>
</tr>
</tbody>
</table>

Fig. 2 | A sampling of the book’s examples. (Part 2 of 2.)

- **Audience.** The examples are accessible to computer science, information technology, software engineering and business students in novice-level and intermediate-level C++ courses. The book is also used by professional programmers.

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

- **Interesting, entertaining and challenging exercises.** Each chapter concludes with a substantial set of exercises, including simple recall of important terminology and concepts, identifying the errors in code samples, writing individual program statements, writing small portions of C++ classes and member and non-member functions, writing complete programs and implementing major projects. Figure 3 lists a sampling of the book’s exercises, including our Making a Difference exercises, which encourage you to use computers and the Internet to research and solve significant social problems. We hope you’ll approach these exercises with your own values, politics and beliefs.
Illustrations and figures. Abundant tables, line drawings, UML diagrams, programs and program outputs are included. A sampling of the book’s drawings and diagrams is shown in (Fig. 4).

### Exercises

<table>
<thead>
<tr>
<th>Card Shuffling and Dealing</th>
<th>Eight Queens</th>
<th>Pythagorean Triples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer-Assisted Instruction</td>
<td>Emergency Response</td>
<td>Salary Calculator</td>
</tr>
<tr>
<td>Computer-Assisted Instruction: Difficulty Levels</td>
<td>Enforcing Privacy with Cryptography</td>
<td>Sieve of Eratosthenes</td>
</tr>
<tr>
<td>Computer-Assisted Instruction: Monitoring Student Performance</td>
<td>Facebook User Base Growth</td>
<td>Simple Decryption</td>
</tr>
<tr>
<td>Computer-Assisted Instruction: Reducing Student Fatigue</td>
<td>Fibonacci Series</td>
<td>Simple Encryption</td>
</tr>
<tr>
<td>Computer-Assisted Instruction: Varying the Types of Problems</td>
<td>Gas Mileage</td>
<td>SMS Language</td>
</tr>
<tr>
<td>Cooking with Healthier Ingredients</td>
<td>Global Warming Facts Quiz</td>
<td>Spam Scanner</td>
</tr>
<tr>
<td>Craps Game Modification</td>
<td>Guess the Number Game</td>
<td>Spelling Checker</td>
</tr>
<tr>
<td>Credit Limits</td>
<td>Hangman Game</td>
<td>Target-Heart-Rate Calculator</td>
</tr>
<tr>
<td>Crossword Puzzle Generator</td>
<td>Health Records Knight’s Tour</td>
<td>Tax Plan Alternatives; The “Fair Tax”</td>
</tr>
<tr>
<td>Cryptograms</td>
<td>Limericks</td>
<td>Telephone number word generator</td>
</tr>
<tr>
<td>De Morgan’s Laws</td>
<td>Maze Traversal: Generating Mazes Randomly</td>
<td>“The Twelve Days of Christmas” Song</td>
</tr>
<tr>
<td>Dice Rolling</td>
<td>Morse Code</td>
<td>Tortoise and the Hare Simulation</td>
</tr>
<tr>
<td></td>
<td>Payroll System</td>
<td>Towers of Hanoi</td>
</tr>
<tr>
<td></td>
<td>Peter Minuit Problem</td>
<td>World Population Growth</td>
</tr>
<tr>
<td></td>
<td>Phishing Scanner</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pig Latin</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Polymorphic Banking Program</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Using Account Hierarchy</td>
<td></td>
</tr>
</tbody>
</table>

#### Main text drawings and diagrams

- **Data hierarchy**
- **Compilation and linking process for multiple source file programs**
- **Order in which a second-degree polynomial is evaluated**
- **GradeBook class diagrams**
- **if single-selection statement activity diagram**
- **if...else double-selection statement activity diagram**
- **while repetition statement**
- **UML activity diagram**
- **for repetition statement UML activity diagram**
- **do...while repetition statement**
- **UML activity diagram**
- **switch multiple-selection statement activity diagram**
- **C++’s single-entry/single-exit sequence, selection and repetition statements**
- **Pass-by-value and pass-by-reference analysis of a program**
- **Inheritance hierarchy diagrams**
- **Function-call stack and activation records**
- **Recursive calls to function fibonacci**
- **Pointer arithmetic diagrams**
- **CommunityMember Inheritance hierarchy**
- **Shape inheritance hierarchy**

### Drawings and diagrams

#### Main text drawings and diagrams

- **Data hierarchy**
- **Compilation and linking process for multiple source file programs**
- **Order in which a second-degree polynomial is evaluated**
- **GradeBook class diagrams**
- **if single-selection statement activity diagram**
- **if...else double-selection statement activity diagram**
- **while repetition statement**
- **UML activity diagram**
- **for repetition statement UML activity diagram**
- **do...while repetition statement**
- **UML activity diagram**
- **switch multiple-selection statement activity diagram**
- **C++’s single-entry/single-exit sequence, selection and repetition statements**
- **Pass-by-value and pass-by-reference analysis of a program**
- **Inheritance hierarchy diagrams**
- **Function-call stack and activation records**
- **Recursive calls to function fibonacci**
- **Pointer arithmetic diagrams**
- **CommunityMember Inheritance hierarchy**
- **Shape inheritance hierarchy**
Other Features

Drawings and diagrams

- public, protected and private inheritance
- Employee hierarchy UML class diagram
- How virtual function calls work
- Stream-I/O template hierarchy
- Two self-referential class objects linked together
- Graphical representation of a list
- Operation insertAtFront represented graphically
- Operation insertAtBack represented graphically
- Operation removeFromFront represented graphically
- Operation removeFromBack represented graphically
- Circular, singly linked list
- Doubly linked list
- Circular, doubly linked list
- Graphical representation of a binary tree

ATM Case Study drawings and diagrams

- Use case diagram for the ATM system from the User’s perspective
- Class diagram showing an association among classes
- Class diagram showing composition relationships
- Class diagram for the ATM system model
- Classes with attributes
- State diagram for the ATM
- Activity diagram for a Balance-Inquiry transaction
- Activity diagram for a withdrawal transaction
- Classes in the ATM system with attributes and operations
- Communication diagram of the ATM executing a balance inquiry
- Communication diagram for executing a balance inquiry
- Sequence diagram that models a withdrawal executing
- Use case diagram for a modified version of our ATM system that also allows users to transfer money between accounts
- Class diagram showing composition relationships of a class
- Class diagram for the ATM system model including class
- Activity diagram for a Deposit transaction
- Activity diagram for a Deposit executing

Fig. 4  A sampling of the book’s drawings and diagrams. (Part 2 of 2.)

- **VideoNotes.** The Companion Website includes many hours of VideoNotes in which co-author Paul Deitel explains in detail key programs in the core chapters. We’ve created a jump table that maps each VideoNote to the corresponding figures in the book (www.deitel.com/books/cpphtp9/jump_table.pdf).

Other Features

- **Pointers.** We provide thorough coverage of the built-in pointer capabilities and the intimate relationship among built-in pointers, C strings and built-in arrays.
- **Visual presentation of searching and sorting, with a simple explanation of Big O.**
- **Printed book contains core content; additional content is online.** A few online chapters and appendices are included. These are available in searchable PDF format on the book’s password-protected Companion Website—see the access card information on the inside front cover.
- **Debugger appendices.** We provide three debugger appendices on the book’s Companion Website—Appendix H, Using the Visual Studio Debugger, Appendix I, Using the GNU C++ Debugger and Appendix J, Using the Xcode Debugger.
Secure C++ Programming

It’s difficult to build industrial-strength systems that stand up to attacks from viruses, worms, and other forms of “malware.” Today, via the Internet, such attacks can be instantaneous and global in scope. Building security into software from the beginning of the development cycle can greatly reduce vulnerabilities.

The CERT® Coordination Center (www.cert.org) was created to analyze and respond promptly to attacks. CERT—the Computer Emergency Response Team—is a government-funded organization within the Carnegie Mellon University Software Engineering Institute™. CERT publishes and promotes secure coding standards for various popular programming languages to help software developers implement industrial-strength systems that avoid the programming practices that leave systems open to attacks.

We’d like to thank Robert C. Seacord, Secure Coding Manager at CERT and an adjunct professor in the Carnegie Mellon University School of Computer Science. Mr. Seacord was a technical reviewer for our book, C How to Program, 7/e, where he scrutinized our C programs from a security standpoint, recommending that we adhere to the CERT C Secure Coding Standard.

We’ve done the same for C++ How to Program, 9/e, adhering to the CERT C++ Secure Coding Standard, which you can find at:

www.securecoding.cert.org

We were pleased to discover that we’ve already been recommending many of these coding practices in our books. We upgraded our code and discussions to conform to these practices, as appropriate for an introductory/intermediate-level textbook. If you’ll be building industrial-strength C++ systems, consider reading Secure Coding in C and C++, Second Edition (Robert Seacord, Addison-Wesley Professional).

Online Content

The book’s Companion Website, which is accessible at

www.pearsonhighered.com/deitel

(see the inside front cover of the book for an access code) contains the following chapters and appendices in searchable PDF format:

- Chapter 24, C++11 Additional Topics
- 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
- 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
- Appendix J, Using the Xcode Debugger
- Appendix K, Test Driving a C++ Program on Mac OS X. (The test drives for Windows and Linux are in Chapter 1.)
The Companion Website also includes:

- Extensive VideoNotes—watch and listen as co-author Paul Deitel discusses key code examples in the core chapters of the book.
- Building Your Own Compiler exercise descriptions from Chapter 19 (posted at the Companion Website and at www.deitel.com/books/cpphtp9).
- Chapter 1 test-drive for Mac OS X.

**Dependency Chart**

The chart in Fig. 6 shows the dependencies among the chapters to help instructors plan their syllabi. *C++ How to Program, 9/e* is appropriate for CS1 and many CS2 courses. The chart shows the book’s modular organization.

**Teaching Approach**

*C++ How to Program, 9/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 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, to ensure that it’s correct 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 key code segments.

*Using fonts for emphasis.* We color the defining occurrence of each key term 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;).

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

*Programming tips.* We include programming tips to help you focus on key aspects of program development. These tips and practices represent the best we’ve gleaned from a combined seven decades of teaching and industry 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.*
1. Most of Chapter 13 is readable after Chapter 7. A small portion requires Chapters 11 and 18.
Summary Bullets. We present a section-by-section, bullet-list summary of the chapter. We include the page number of each term’s defining occurrence in the chapter for easy reference.

Index. We’ve included an extensive index, with defining occurrences of key terms highlighted with a bold blue page number.

Obtaining the Software Used in C++ How to Program, 9/e

We wrote the code examples in C++ How to Program, 9/e using the following C++ development tools:

- Microsoft’s free Visual Studio Express 2012 for Windows Desktop, which includes Visual C++ and other Microsoft development tools. This runs on Windows 7 and 8 and is available for download at

  www.microsoft.com/visualstudio/eng/downloads#
d-express-windows-desktop

- GNU’s free GNU C++ (gcc.gnu.org/install/binaries.html), which is already installed on most Linux systems and can also be installed on Mac OS X and Windows systems.

- Apple’s free Xcode, which OS X users can download from the Mac App Store.

Instructor Supplements

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

- Solutions Manual contains solutions to most of the end-of-chapter exercises. We’ve added many Making a Difference exercises, most with solutions. Please do not write to us requesting access to the Pearson Instructor’s Resource Center. Access is restricted to college instructors teaching from the book. Instructors may obtain access only through their Pearson representatives. If you’re not a registered faculty member, contact your Pearson representative or visit www.pearsonhighered.com/educator/repllocator/. Exercise Solutions are not provided for “project” exercis-
Preface

Check out our Programming Projects Resource Center for lots of additional exercise and project possibilities

www.deitel.com/ProgrammingProjects

- 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

Online Practice and Assessment with MyProgrammingLab™

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

A self-study and homework tool, a MyProgrammingLab course consists of hundreds of small practice problems organized around the structure of this textbook. For students, the system automatically detects errors in the logic and syntax of their code submissions and offers targeted hints that enable students to figure out what went wrong—and why. For instructors, a comprehensive gradebook tracks correct and incorrect answers and stores the code inputted by students for review.

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

Acknowledgments

We’d like to thank Abbey Deitel and Barbara Deitel of Deitel & Associates, Inc. for long hours devoted to this project. Abbey co-authored Chapter 1 and she and Barbara painstakingly researched the new capabilities of C++11.

We’re fortunate to have worked with the dedicated team of publishing professionals at Pearson Higher Education. We appreciate the guidance, wisdom and energy of Tracy Johnson, Executive Editor, Computer Science. Carole Snyder did an extraordinary job recruiting the book’s reviewers and managing the review process. Bob Engelhardt did a wonderful job bringing the book to publication.

Reviewers

We wish to acknowledge the efforts of our reviewers. The book was scrutinized by current and former members of the C++ standards committee that developed C++11, academics teaching C++ courses and industry experts. They provided countless suggestions for improving the presentation. Any remaining flaws in the book are our own.

Ninth Edition reviewers: Dean Michael Berris (Google, Member ISO C++ Committee), Danny Kalev (C++ expert, certified system analyst and former member of the C++ Standards Committee), Linda M. Krause (Elmhurst College), James P. McNellis (Microsoft Corporation), Robert C. Seacord (Secure Coding Manager at SEI/CERT, author of Secure Coding in C and C++) and José Antonio González Seco (Parliament of Andalusia).

Other recent edition reviewers: Virginia Bailey (Jackson State University), Thomas J. Borrelli (Rochester Institute of Technology), Ed Brey (Kohler Co.), 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), Anne B. Horton (Lockheed Martin), Terrell Hull (Logicalis Integration Solutions), Ed James-Beckham (Borland), Wing-Ning Li (University of Arkansas), Dean Mathias (Utah State University), Robert A. McLain (Tidewater Community College), Robert Myers (Florida State University), Gavin Osborne (Saskatchewan Inst. of App. Sci. and Tech.), Amar Raheja (California State Polytechnic University, Pomona), April Reagan (Microsoft), Raymond Stephenson (Microsoft), Dave Topham (Ohlone College), Anthony Williams (author and C++ Standards Committee member) and Chad Willwerth (University Washington, Tacoma).

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

diteit@deitel.com

We’ll respond promptly. We enjoyed writing *C++ How to Program, Ninth Edition*. We hope you enjoy reading it!

Paul Deitel
Harvey Deitel

**About the Authors**

**Paul 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 programming courses to industry clients, including Cisco, IBM, Siemens, Sun Microsystems, Dell, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, White Sands Missile Range, Rogue Wave Software, Boeing, SunGard Higher Education, 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/professional book/video authors.

**Dr. Harvey 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 in Electrical Engineering from MIT and a Ph.D. in Mathematics from Boston University. 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., in 1991 with his son, Paul Deitel. The Deitels’ publications have earned international recognition, with translations published in Chinese, Korean, Japanese, German, Russian, Spanish, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to corporate, academic, government and military clients.

**Corporate Training from Deitel & Associates, Inc.**

Deitel & Associates, Inc., founded by Paul Deitel and Harvey Deitel, is an internationally recognized authoring and corporate training organization, specializing in computer programming languages, object technology, mobile app development and Internet and web
software technology. The company’s clients include many of the world’s largest compa-
nies, government agencies, branches of the military, and academic institutions. The com-
pany offers instructor-led training courses delivered at client sites worldwide on major
programming languages and platforms, including C++, Visual C++®, C, Java™, Visual
C#®, Visual Basic®, XML®, Python®, object technology, Internet and web program-
ming, Android app development, Objective-C and iPhone app development and a grow-
ing list of additional programming and software development courses.

Through its 36-year publishing partnership with Prentice Hall/Pearson, Deitel &
Associates, Inc., publishes leading-edge programming college textbooks, professional
books and LiveLessons video courses. Deitel & Associates, Inc. and the authors can be
reached at:

deitel@deitel.com

To learn more about Deitel’s Dive-Into® Series Corporate Training curriculum, visit:

www.deitel.com/training

To request a proposal for worldwide on-site, instructor-led training at your organization,
e-mail deitel@deitel.com.

Individuals wishing to purchase Deitel books and LiveLessons video training 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/information/index.page