Preface

"Be faithful in small things because it is in them that your strength lies."
—Mother Teresa

Welcome to C++ programming and *Small C++ How to Program, Fifth Edition*! C++ is a world-class programming language for developing industrial-strength, high-performance computer applications. At Deitel & Associates, we write college-level computer science textbooks and professional reference books. *Small C++ How to Program, Fifth Edition* was a joy to create. Our goal was to design a smaller, lower-priced book for one-semester introductory (CS1) courses based on the major revision of our book *C++ How to Program, Fifth Edition*. *Small C++ How to Program, Fifth Edition* focuses on the core concepts and features of C++ covered in Chapters 1–13 of *C++ How to Program, 5/e*.

We believe that this book and its support materials have everything instructors and students need for an informative, interesting, challenging and entertaining C++ educational experience. In this Preface, we overview the many features of *Small C++ How to Program, 5/e*. The *Tour of the Book* section of the Preface gives instructors, students and professionals a sense of *Small C++ How to Program, 5/e*'s coverage of C++ and object-oriented programming. We also overview various conventions used in the book, such as syntax coloring the code examples, "code washing" and code highlighting. We provide information about free compilers available on the Web. We also discuss the comprehensive suite of educational materials that help instructors maximize their students' learning experience, including the *Instructor's Resource CD*, PowerPoint® Slide lecture notes, course management systems, SafariX (Pearson Education's WebBook publications) and more.

Features of *Small C++ How to Program, 5/e*

To create *Small C++ How to Program, 5/e*, we put the previous edition of *C++ How to Program* under the microscope. The new edition has many compelling features:

- **Major Content Revisions.** All the chapters have been significantly updated and upgraded. We tuned the writing for clarity and precision. We also adjusted our use of C++ terminology in accordance with the ANSI/ISO C++ standard document that defines the language.
- **Smaller Chapters.** Larger chapters have been divided into smaller, more manageable chapters (e.g., Chapter 1 of the Fourth Edition has been split into Chapters 1–2; Chapter 2 of the Fourth Edition is now Chapters 4–5).
- **Early Classes and Objects Approach.** We changed to an early classes and objects pedagogy. Students are introduced to the basic concepts and terminology of object technology in Chapter 1. In the previous edition, students began developing customized, reusable classes and objects in Chapter 6, but in this edition, they do

so in our completely new Chapter 3. Chapters 4–7 have been carefully rewritten from an "early classes and objects" perspective. This new edition is object oriented, where appropriate, from the start and throughout the text. Moving the discussion of objects and classes to earlier chapters 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.

- **Integrated Case Studies.** We added several case studies spanning multiple sections and chapters that often build on a class introduced earlier in the book to demonstrate new programming concepts taught later in the book. These case studies include the development of the `GradeBook` class in Chapters 3–7, the `Time` class in several sections of Chapters 9–10 and the `Employee` class in Chapters 12–13.
- **Integrated `GradeBook` Case Study.** We have added a new `GradeBook` case study to reinforce our early classes and objects presentation. The 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.
- **Essential Topics for CS1.** *Small C++ How to Program, 5/e* focuses on core C++ concepts presented in CS1 courses. It is designed for first courses in computing and is appropriate for Computer Science and Information Systems courses. The book is information-rich enough for the first two courses at many schools.
- **Unified Modeling Language™ 2.0 (UML 2.0)—Introducing the UML 2.0.** The Unified Modeling Language (UML) has become the preferred graphical modeling language for designing object-oriented systems. All the UML diagrams in the book comply with the new UML 2.0 specification. We use UML class diagrams to visually represent classes and their inheritance relationships, and we use UML activity diagrams to demonstrate the flow of control in each of C++'s control statements.
- **Compilation and Linking Process for Multiple-Source-File Programs.** Chapter 3 includes a diagram and discussion of the compilation and linking process that produces an executable application.
- **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.
- **Early Introduction of C++ Standard Library `string` and `vector` Objects.** 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 8, 10, 11 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 assignments in industry where they will work 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. We continue to discuss C-like pointer-based arrays in Chapter 7 to prepare students for working with C and C++ legacy code in industry and to use as a basis for building our own customized `Array` class in Chapter 11, Operator Overloading; String and Array Objects.
- **Tuned Treatment of Inheritance and Polymorphism.** Chapters 12–13 have been carefully tuned, making the treatment of inheritance and polymorphism clearer and more accessible for students who are new to OOP. An `Employee` hierarchy replaces the `Point/Circle/Cylinder` hierarchy used in prior editions to introduce inheritance and polymorphism. This new hierarchy is more natural.
- **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 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.
- **ANSI/ISO C++ Standard Compliance.** We have audited our presentation against the most recent ANSI/ISO C++ standard document for completeness and accuracy. [*Note:* If you need additional technical details on C++, you may want to read the C++ standard document. An electronic PDF copy of the C++ standard document, number INCITS/ISO/IEC 14882-2003, is available for $18 at `webstore.ansi.org/ansidocstore/default.asp`.]
- **New Debugger Appendices.** We include two new "Using the Debugger" appendices: Appendix F, Using the Visual C++ .NET Debugger, and Appendix G, Using the GNU C++ Debugger.
- To keep this book small and focused on first courses, we removed the optional ten-section OOD/UML case study that appears in *C++ How to Program, 5/e*.
- **New Interior Design.** Working with the creative services team at Prentice Hall, we redesigned the interior styles for our *How to Program* Series. The new fonts are easier on the eyes and the new art package is more appropriate for the more detailed illustrations. We now place the defining occurrence of each key term both in the text and in the index in **blue, bold style** text for easier reference. We emphasize on-screen components in the bold **Helvetica** font (e.g., the **File** menu) and emphasize C++ program text in the `Lucida` font (e.g., `int x = 5`).
- **Syntax Coloring.** We syntax color all the C++ code, which is consistent with most C++ integrated development environments and code editors. This greatly improves code readability—an especially important goal, given that this book contains 8,720 lines of code. Our syntax-coloring conventions are as follows:

```
comments appear in green
keywords appear in dark blue
constants and literal values appear in light blue
errors appear in red
all other code appears in black
```

- **Code Highlighting.** Extensive code highlighting makes it easy for readers to locate each program's new features and helps students review the material rapidly when preparing for exams or labs.
- "**Code washing.**" This is our term for using extensive and meaningful comments, using meaningful identifiers, applying uniform indentation conventions, aligning curly braces vertically, using a `// end...` comment on every line with a right curly brace and using vertical spacing to highlight significant program units such as control statements and functions. This process results in programs that are easy to read and self-documenting. We have extensively "code washed" all the source-code programs in both the text and the book's ancillaries. We have worked hard to make our code exemplary.
- **Code Testing on Multiple Platforms.** We tested the code examples on various popular C++ platforms. For the most part, all of the examples in this book port easily to all popular ANSI/ISO standard-compliant compilers. We will post any problems at `www.deitel.com/books/scpphtp5/index.html`.
- **Errors and Warnings Shown for Multiple Platforms.** For programs that intentionally contain errors to illustrate a key concept, we show the error messages that result on several popular platforms.
- **Large Review Team.** The book has been carefully scrutinized by a team of 21 distinguished academic and industry reviewers.
- **Free Web-Based *Cyber Classroom*.** We've converted our popular interactive multimedia version of the text (which we call a *Cyber Classroom*) from a for-sale, CD-based product to a free online supplement, available with new books purchased from Prentice Hall for fall 2005 classes.
- **Free Student Solutions Manual.** We've converted our Student Solutions Manual, which contains solutions to approximately half of the exercises, from a for-sale softcover book to a free online supplement, available with new books purchased from Prentice Hall for fall 2005 classes.
- **Free Lab Manual.** We've converted our Lab Manual, *C++ in the Lab*, from a for-sale softcover book to a free online supplement included with the *Cyber Classroom*, available with new books purchased from Prentice Hall for fall 2005 classes.

As you read this book, if you have questions, send an e-mail to `deitel@deitel.com`; we will respond promptly. Please visit our Web site, `www.deitel.com` and be sure to sign up for the free *DEITEL® Buzz Online* e-mail newsletter at `www.deitel.com/newsletter/subscribe.html` for updates to this book and the latest information on C++. We also use the Web site and the newsletter to keep our readers and industry clients informed of the latest news on Deitel publications and services. Please check the following Web site regularly for errata, updates regarding the C++ software, free downloads and other resources:

```
www.deitel.com/books/scpphtp5/index.html
```

Teaching Approach

Small C++ How to Program, 5/e contains an abundant collection of examples, exercises and projects drawn from many fields to provide the student with a chance to solve interesting

real-world problems. The book concentrates on the principles of good software engineering and stresses program clarity. We avoid arcane terminology and syntax specifications in favor of teaching by example. We are educators who teach programming languages courses in industry classrooms worldwide. Dr. Harvey M. Deitel has 20 years of college teaching experience, including serving as chairman of the Computer Science Department at Boston College, and 15 years of industry teaching experience. Paul Deitel has 12 years of industry teaching experience. The Deitels have taught C++ courses at all levels to the government, industry, military and academic clients of Deitel & Associates.

Learning C++ using the LIVE-CODE Approach

Small C++ How to Program, 5/e, is loaded with C++ programs—each new concept is presented in the context of a complete working C++ program that is immediately followed by one or more sample executions showing the program's inputs and outputs. This style exemplifies the way we teach and write about programming. We call this method of teaching and writing the LIVE-CODE Approach. *We use programming languages to teach programming languages.* Reading the examples in the text is much like typing and running them on a computer. We provide all the source code for the book's examples at `www.deitel.com` and on the accompanying CD—making it easy for students to run each example as they study it.

World Wide Web Access

All the source-code examples for *Small C++ How to Program, 5/e,* (and our other publications) are available on the Internet as downloads from

```
www.deitel.com
```

Registration is quick and easy, and the downloads are free. We suggest downloading all the examples (or copying them from the CD included in the back of this book), then running each program as you read the corresponding text. Making changes to the examples and immediately seeing the effects of those changes is a great way to enhance your C++ learning experience.

Objectives

Each chapter begins with a statement of objectives. This lets students know what to expect and gives them an opportunity, after reading the chapter, to determine if they have met these objectives. This is a confidence builder and a source of positive reinforcement.

Quotations

The learning objectives are followed by quotations. Some are humorous, some philosophical and some offer interesting insights. We hope that you will enjoy relating the quotations to the chapter material. Many of the quotations are worth a second look after reading the chapter.

Outline

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

8,720 Lines of Syntax-Colored Code in 124 Example Programs with Live Sample Program Inputs and Outputs

Our LIVE-CODE programs range in size from just a few lines of code to substantial larger examples. Each program is followed by a window containing the input/output dialogue

produced when the program is run, so students can confirm that the programs run as expected. Relating outputs to the program statements that produce them is an excellent way to learn and to reinforce concepts. Our programs demonstrate the diverse features of C++. The code is line-numbered and syntax colored—with C++ keywords, comments and other program text appearing in different colors. This facilitates reading the code—students will especially appreciate the syntax coloring when they read the larger programs.

402 Illustrations/Figures

An abundance of charts, tables, line drawings, programs and program outputs is included. We model the flow of control in control statements with UML activity diagrams. UML class diagrams model the data members, constructors and member functions of classes.

403 Programming Tips

We include programming tips to help students focus on important aspects of program development. We highlight these tips in the form of *Good Programming Practices*, *Common Programming Errors*, *Performance Tips*, *Portability Tips*, *Software Engineering Observations* and *Error-Prevention Tips*. 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, lemmas and corollaries in mathematics books—it provides a basis on which to build good software.

Good Programming Practices

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

Common Programming Errors

Students who are new to programming (or a programming language) tend to make certain errors frequently. Focusing on these Common Programming Errors *reduces the likelihood that students will make the same mistakes and shortens long lines outside instructors' offices during office hours!*

Performance Tips

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

Portability Tips

Software development is a complex and expensive activity. Organizations that develop software must often produce versions customized to a variety of computers and operating systems. So there is a strong emphasis today on portability, i.e., on producing software that will run on a variety of computer systems with few, if any, changes. Some programmers assume that if they implement an application in standard C++, the application will be portable. This simply is not the case. Achieving portability requires careful and cautious design. There are many pitfalls. We include Portability Tips *to help students write portable code and to provide insights on how C++ achieves its high degree of portability.*

Software Engineering Observations

The object-oriented programming paradigm necessitates a complete rethinking of the way we build software systems. C++ is an effective language for achieving good software engineering. The Software Engineering Observations *highlight architectural and design issues that affect the construction of software systems, especially large-scale systems. Much of what the student learns here will be useful in upper-level courses and in industry as the student begins to work with large, complex real-world systems.*

Error-Prevention Tips

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

Wrap-Up Sections
Each chapter ends with additional pedagogical devices. New in this edition, each chapter ends with a brief "wrap-up" section that recaps the topics that were presented. The wrap-ups also help the student transition to the next chapter.

Summary (567 Summary bullets)
We present a thorough, bullet-list-style summary at the end of every chapter. On average, there are 43 summary bullets per chapter. This focuses the student's review and reinforces key concepts.

Terminology (969 Terms)
We include an alphabetized list of the important terms defined in each chapter—again, for further reinforcement. There is an average of 64 terms per chapter. Each term also appears in the index, and the defining occurrence of each term is highlighted in the index with a **blue, bold** page number so the student can locate the definitions of terms quickly.

333 Self-Review Exercises and Answers (Count Includes Separate Parts)
Extensive self-review exercises and answers are included for self-study. This gives the student a chance to build confidence with the material and prepare for the regular exercises. We encourage students to do all the self-review exercises and check their answers.

543 Exercises (Solutions in Instructor's Manual; Count Includes Separate Parts)
Each chapter concludes with a substantial set of exercises including simple recall of important terminology and concepts; writing individual C++ statements; writing small portions of C++ functions and classes; writing complete 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 audiences and to vary course assignments each semester. Instructors can use these exercises to form homework assignments, short quizzes and major examinations. The solutions for the vast majority of the exercises are included on the *Instructor's Resource CD (IRCD),* which is *available only to instructors* through their Prentice Hall representatives. [**NOTE: Please do not write to us requesting the Instructor's CD. Distribution of this ancillary is limited strictly to college instructors teaching from the book. Instructors may obtain the solutions manual only from their Prentice Hall representatives.**] Students will have access to approximately half the exercises in the book in the free, Web-based *Cyber Classroom* which will be available in late spring 2005. For more in-

formation about the *Cyber Classroom*, please visit `www.deitel.com` or sign up for the free *Deitel® Buzz Online* e-mail newsletter at `www.deitel.com/newsletter/subscribe.html`.

Approximately 2900 Index Entries
We have included an extensive index. This helps students find terms or concepts by keyword. The index is useful to people reading the book for the first time and is especially useful to practicing programmers who use the book as a reference.

"Double Indexing" of All C++ LIVE-CODE Examples
Small C++ How to Program, 5/e has 124 live-code examples and 543 exercises (including separate parts). We have double-indexed each of the live-code examples and most of the more substantial exercises. For every source-code program in the book, we indexed the figure caption both alphabetically and as a subindex item under "Examples." This makes it easier to find examples that include particular features. The more substantial exercises are also indexed both alphabetically and as subindex items under "Exercises."

Tour of the Book

In this section, we take a tour of the many capabilities of C++ you will study in *Small C++ How to Program, 5/e*. Note that each of the chapters in this book depends on the preceding chapters except Chapter 11, Operator Overloading; String and Array Objects, which can be skipped and covered after either Chapter 12 or 13.

Chapter 1—Introduction to Computers, the Internet and World Wide Web— discusses what computers are, how they work and how they are programmed. The chapter gives a brief history of the development of programming languages from machine languages to assembly languages and high-level languages. The origin of the C++ programming language is discussed. The chapter includes an introduction to a typical C++ programming environment. We walk readers through a "test drive" of a typical C++ application on Windows and Linux systems. Our free *Dive-Into™ Series* publications, which discuss compiling and running C++ applications on various platforms, are available for download at `www.deitel.com/books/cpphtp5/index.html`. Chapter 1 concludes with an introduction to basic object technology concepts and terminology and the Unified Modeling Language.

Chapter 2—Introduction to C++ Programming—provides a lightweight introduction to programming applications in the C++ programming language. The chapter introduces nonprogrammers to basic programming concepts and constructs. The programs in this chapter illustrate how to display data on the screen and how to obtain data from the user at the keyboard. Chapter 2 ends with detailed treatments of decision making and arithmetic operations.

Chapter 3—Introduction to Classes and Objects—is the "featured" chapter for the new edition. It provides a friendly early introduction to classes and objects. Carefully developed and completely new in this edition, Chapter 3 gets students working with object orientation comfortably from the start. It was developed with the guidance of a distinguished team of industry and academic reviewers. We introduce classes, objects, member functions, constructors and data members using a series of simple real-world examples. We develop a well-engineered framework for organizing object-oriented programs in C++. First, we motivate the notion of classes with a simple example. Then we

present a carefully paced sequence of seven complete working programs to demonstrate creating and using your own classes. These examples begin our **integrated case study on developing a grade-book class** that instructors can use to maintain student test scores. This case study is enhanced over the next several chapters, culminating with the version presented in Chapter 7, Arrays and Vectors. The **`GradeBook` class case study** describes how to define a class and how to use it to create an object. The case study discusses how to declare and define member functions to implement the class's behaviors, how to declare data members to implement the class's attributes and how to call an object's member functions to make them perform their tasks. We introduce C++ Standard Library class `string` and create `string` objects to store the name of the course that a `GradeBook` object represents. Chapter 3 explains the differences between data members of a class and local variables of a function and how to use a constructor to ensure that an object's data is initialized when the object is created. We show how to promote software reusability by separating a class definition from the client code (e.g., function `main`) that uses the class. We also introduce another fundamental principle of good software engineering—separating interface from implementation. The chapter includes a detailed diagram and discussion explaining the compilation and linking process that produces an executable application.

Chapter 4—Control Statements: Part 1—focuses on the program-development process involved in creating useful classes. The chapter discusses how to take a problem statement and develop a working C++ program from it, including performing intermediate steps in pseudocode. The chapter introduces some simple control statements for decision making (`if` and `if...else`) and repetition (`while`). We examine counter-controlled and sentinel-controlled repetition using the **`GradeBook` class** from Chapter 3, and introduce C++'s increment, decrement and assignment operators. The chapter includes **two enhanced versions of the `GradeBook` class**, each based on Chapter 3's final version. These versions each include a member function that uses control statements to calculate the average of a set of student grades. In the first version, the member function uses counter-controlled repetition to input 10 student grades from the user, then determines the average grade. In the second version, the member function uses sentinel-controlled repetition to input an arbitrary number of grades from the user, then calculates the average of the grades that were entered. The chapter uses simple UML activity diagrams to show the flow of control through each of the control statements.

Chapter 5—Control Statements: Part 2—continues the discussion of C++ control statements with examples of the `for` repetition statement, the `do...while` repetition statement, the `switch` selection statement, the `break` statement and the `continue` statement. We create an **enhanced version of class `GradeBook`** that uses a `switch` statement to count the number of A, B, C, D and F grades entered by the user. This version uses sentinel-controlled repetition to input the grades. While reading the grades from the user, a member function modifies data members that keep track of the count of grades in each letter grade category. Another member function of the class then uses these data members to display a summary report based on the grades entered. The chapter includes a discussion of logical operators.

Chapter 6—Functions and an Introduction to Recursion—takes a deeper look inside objects and their member functions. We discuss C++ standard-library functions and examine more closely how students can build their own functions. The techniques presented in Chapter 6 are essential to the production of properly organized programs, espe-

cially the kinds of larger programs and software that system programmers and application programmers are likely to develop in real-world applications. The "divide and conquer" strategy is presented as an effective means for solving complex problems by dividing them into simpler interacting components. The chapter's first example continues the **`GradeBook` class case study** with an example of a function with multiple parameters. Students enjoy the chapter's treatment of random numbers and simulation, and the discussion of the dice game of craps, which makes elegant use of control statements. The chapter discusses the so-called "C++ enhancements to C," including `inline` functions, reference parameters, default arguments, the unary scope resolution operator, function overloading and function templates. We also present C++'s call-by-value and call-by-reference capabilities. The header files table introduces many of the header files that the reader will use throughout the book. In this new edition, 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. The chapter offers a solid introduction to recursion and includes a table summarizing the recursion examples and exercises distributed throughout the remainder of the book. Some texts leave recursion for a chapter late in the book; we feel this topic is best covered gradually throughout the text. The extensive collection of exercises at the end of the chapter includes several classic recursion problems including the Towers of Hanoi.

Chapter 7—Arrays and Vectors—explains how to process lists and tables of values. We discuss the structuring of data in arrays of data items of the same type and demonstrate how arrays facilitate the tasks performed by objects. The early sections of this chapter use C-style, pointer-based arrays, which, as you will see in Chapter 8, are really pointers to the array contents in memory. We then present arrays as full-fledged objects in the last section of the chapter, where we introduce the C++ Standard Library `vector` class template—a robust array data structure. The chapter presents numerous examples of both one-dimensional arrays and two-dimensional arrays. Examples in the chapter investigate various common array manipulations, printing bar charts, sorting data and passing arrays to functions. The chapter includes the **final two `GradeBook` case study sections**, in which we use arrays to store student grades for the duration of a program's execution. Previous versions of the class process a set of grades entered by the user, but do not maintain the individual grade values in data members of the class. In this chapter, we use arrays to enable an object of the `GradeBook` class to maintain a set of grades in memory, thus eliminating the need to repeatedly input the same set of grades. The first version of the class stores the grades in a one-dimensional array and can produce a report containing the average of the grades, the minimum and maximum grades and a bar chart representing the grade distribution. The second version (i.e., the final version in the case study) uses a two-dimensional array to store the grades of a number of students on multiple exams in a semester. This version can calculate each student's semester average, as well as the minimum and maximum grades across all grades received for the semester. The class also produces a bar chart displaying the overall grade distribution for the semester. Another key feature of this chapter is the discussion of elementary sorting and searching techniques. The end-of-chapter exercises include a variety of interesting and challenging problems, such as improved sorting techniques, the design of a simple airline reservations system, an introduction to the concept of turtle graphics (made famous in the LOGO language) and the Knight's Tour and Eight Queens

problems that introduce the notion of heuristic programming widely employed in the field of artificial intelligence. The exercises conclude with many recursion problems including selection sort, palindromes, linear search, the Eight Queens, printing an array, printing a string backwards and finding the minimum value in an array.

Chapter 8—Pointers and Pointer-Based Strings—presents one of the most powerful features of the C++ language—pointers. The chapter provides detailed explanations of pointer operators, call by reference, pointer expressions, pointer arithmetic, the relationship between pointers and arrays, arrays of pointers and pointers to functions. Chapter 8 demonstrates how to use `const` with pointers to enforce the principle of least privilege to build more robust software. We also introduce and using the `sizeof` operator to determine the size of a data type or data items in bytes during program compilation. There is an intimate relationship between pointers, arrays and C-style strings in C++, so we introduce basic C-style string-manipulation concepts and discuss some of the most popular C-style string-handling functions, such as `getline` (input a line of text), `strcpy` and `strncpy` (copy a string), `strcat` and `strncat` (concatenate two strings), `strcmp` and `strncmp` (compare two strings), `strtok` ("tokenize" a string into its pieces) and `strlen` (return the length of a string). In this new edition, we use `string` objects (introduced in Chapter 3) in place of C-style, `char *` pointer-based strings wherever possible. However, we include `char *` strings in Chapter 8 to help the reader master pointers and prepare for the professional world in which the reader will see a great deal of C legacy code that has been implemented over the last three decades. Thus, the reader will become familiar with the two most prevalent methods of creating and manipulating strings in C++. Many people find that the topic of pointers is, by far, the most difficult part of an introductory programming course. In C and "raw C++," arrays and strings are pointers to array and string contents in memory (even function names are pointers). Studying this chapter carefully should reward you with a deep understanding of pointers. The chapter is loaded with challenging exercises. The chapter exercises include a simulation of the classic race between the tortoise and the hare, card-shuffling and dealing algorithms, recursive quicksort and recursive maze traversals. A special section entitled "Building Your Own Computer" also is included. This section explains machine-language programming and proceeds with a project involving the design and implementation of a computer simulator that leads the student to write and run machine-language programs. This unique feature of the text will be especially useful to the reader who wants to understand how computers really work. Our students enjoy this project and often implement substantial enhancements, many of which are suggested in the exercises. A second special section includes challenging string-manipulation exercises related to text analysis, word processing, printing dates in various formats, check protection, writing the word equivalent of a check amount, Morse Code and metric-to-English conversions.

Chapter 9—Classes: A Deeper Look, Part 1—continues our discussion of object-oriented programming. This chapter uses a rich `Time` class case study to illustrate accessing class members, separating interface from implementation, using access functions and utility functions, initializing objects with constructors, destroying objects with destructors, assignment by default memberwise copy and software reusability. Students learn the order in which constructors and destructors are called during the lifetime of an object. A modification of the `Time` case study demonstrates the problems that can occur when a member function returns a reference to a `private` data member, which breaks the encapsulation of the class. The

chapter exercises challenge the student to develop classes for times, dates, rectangles, and playing tic-tac-toe. Students generally enjoy game-playing programs. Mathematically inclined readers will enjoy the exercises on creating class `Complex` (for complex numbers), class `Rational` (for rational numbers) and class `HugeInteger` (for arbitrarily large integers).

Chapter 10—Classes: A Deeper Look, Part 2—continues the study of classes and presents additional object-oriented programming concepts. The chapter discusses declaring and using constant objects, constant member functions, composition—the process of building classes that have objects of other classes as members, `friend` functions and `friend` classes that have special access rights to the `private` and `protected` members of classes, the `this` pointer, which enables an object to know its own address, dynamic memory allocation, `static` class members for containing and manipulating class-wide data, examples of popular abstract data types (arrays, strings and queues), container classes and iterators. In our discussion of `const` objects, we mention keyword `mutable` which is used in a subtle manner to enable modification of "non-visible" implementation in `const` objects. We discuss dynamic memory allocation using `new` and `delete`. When `new` fails, the program terminates by default because `new` "throws an exception" in standard C++. We motivate the discussion of `static` class members with a video-game-based example. We emphasize how important it is to hide implementation details from clients of a class; then, we discuss proxy classes, which provide a means of hiding implementation (including the `private` data in class headers) from clients of a class. The chapter exercises include developing a saving-account class and a class for holding sets of integers.

Chapter 11—Operator Overloading; String and Array Objects—presents one of the most popular topics in our C++ courses. Students really enjoy this material. They find it a perfect match with the detailed discussion of crafting valuable classes in Chapters 9 and 10. Operator overloading enables the programmer to tell the compiler how to use existing operators with objects of new types. C++ already knows how to use these operators with objects of built-in types, such as integers, floats and characters. But suppose that we create a new `String` class—what would the plus sign mean when used between `String` objects? Many programmers use + with strings to mean concatenation. In Chapter 11, the programmer will learn how to "overload" the plus sign, so when it is written between two `String` objects in an expression, the compiler will generate a function call to an "operator function" that will concatenate the two `String`s. The chapter discusses the fundamentals of operator overloading, restrictions in operator overloading, overloading with class member functions vs. with nonmember functions, overloading unary and binary operators and converting between types. Chapter 11 features a collection of substantial case studies including an array class, a `String` class, a date class, a huge integer class and a complex numbers class (the last two appear with full source code in the exercises). Mathematically inclined students will enjoy creating the `polynomial` class in the exercises. This material is different from most programming languages and courses. Operator overloading is a complex topic, but an enriching one. Using operator overloading wisely helps you add extra "polish" to your classes. The discussions of class `Array` and class `String` are particularly valuable to students who have already used the C++ Standard Library `string` class and `vector` class template that provide similar capabilities. The exercises encourage the student to add operator overloading to classes `Complex`, `Rational` and `HugeInteger` to enable convenient manipulation of objects of these classes with operator symbols—as in mathematics—rather than with function calls as the student did in the Chapter 10 exercises.

Chapter 12—Object-Oriented Programming: Inheritance—introduces one of the most fundamental capabilities of object-oriented programming languages—inheritance: a form of software reusability in which new classes are developed quickly and easily by absorbing the capabilities of existing classes and adding appropriate new capabilities. In the context of an **`Employee` hierarchy** case study, this substantially revised chapter presents a five-example sequence demonstrating `private` data, `protected` data and good software engineering with inheritance. We begin by demonstrating a class with `private` data members and `public` member functions to manipulate that data. Next, we implement a second class with additional capabilities, intentionally and tediously duplicating much of the first example's code. The third example begins our discussion of inheritance and software reuse—we use the class from the first example as a base class and quickly and simply inherit its data and functionality into a new derived class. This example introduces the inheritance mechanism and demonstrates that a derived class cannot access its base class's `private` members directly. This motivates our fourth example, in which we introduce `protected` data in the base class and demonstrate that the derived class can indeed access the `protected` data inherited from the base class. The last example in the sequence demonstrates proper software engineering by defining the base class's data as `private` and using the base class's `public` member functions (that were inherited by the derived class) to manipulate the base class's `private` data in the derived class. The chapter discusses the notions of base classes and derived classes, `protected` members, `public` inheritance, `protected` inheritance, `private` inheritance, direct base classes, indirect base classes, constructors and destructors in base classes and derived classes, and software engineering with inheritance. The chapter also compares inheritance (the *is-a* relationship) with composition (the *has-a* relationship) and introduces the *uses-a* and *knows-a* relationships.

Chapter 13—Object-Oriented Programming: Polymorphism—deals with another fundamental capability of object-oriented programming: polymorphic behavior. The completely revised Chapter 13 builds on the inheritance concepts presented in Chapter 12 and focuses on the relationships among classes in a class hierarchy and the powerful processing capabilities that these relationships enable. When many classes are related to a common base class through inheritance, each derived-class object may be treated as a base-class object. This enables programs to be written in a simple and general manner independent of the specific types of the derived-class objects. New kinds of objects can be handled by the same program, thus making systems more extensible. Polymorphism enables programs to eliminate complex `switch` logic in favor of simpler "straight-line" logic. A screen manager of a video game, for example, can send a `draw` message to every object in a linked list of objects to be drawn. Each object knows how to draw itself. An object of a new class can be added to the program without modifying that program (as long as that new object also knows how to draw itself). The chapter discusses the mechanics of achieving polymorphic behavior via `virtual` functions. It distinguishes between abstract classes (from which objects cannot be instantiated) and concrete classes (from which objects can be instantiated). Abstract classes are useful for providing an inheritable interface to classes throughout the hierarchy. We demonstrate abstract classes and polymorphic behavior by revisiting the **`Employee` hierarchy** of Chapter 12. We introduce an abstract `Employee` base class, from which classes `CommissionEmployee`, `HourlyEmployee` and `SalariedEmployee` inherit directly and class `BasePlusCommissionEmployee` inherits indirectly. In the past, our professional audiences have insisted that we

provide a deeper explanation that shows precisely how polymorphism is implemented in C++, and hence, precisely what execution time and memory "costs" are incurred when programming with this powerful capability. We responded by developing an illustration and a precise explanation of the *vtables* (`virtual` function tables) that the C++ compiler builds automatically to support polymorphism. To conclude, we introduce run-time type information (RTTI) and dynamic casting, which enable a program to determine an object's type at execution time, then act on that object accordingly. Using RTTI and dynamic casting, we give a 10% pay increase to employees of a specific type, then calculate the earnings for such employees. For all other employee types, we calculate their earnings polymorphically.

Appendix A—Operator Precedence and Associativity Chart—presents the complete set of C++ operator symbols, in which each operator appears on a line by itself with its operator symbol, its name and its associativity.

Appendix B—ASCII Character Set—All the programs in this book use the ASCII character set, which is presented in this appendix.

Appendix C—Fundamental Types—lists all fundamental types defined in the *C++ Standard*.

Appendix D—Number Systems—discusses the binary, octal, decimal and hexadecimal number systems. It considers how to convert numbers between bases and explains the one's complement and two's complement binary representations.

Appendix E—C++ Internet and Web Resources—contains a listing of valuable C++ resources, such as demos, information about popular compilers (including "freebies"), books, articles, conferences, job banks, journals, magazines, help, tutorials, FAQs (frequently asked questions), newsgroups, Web-based courses, product news and C++ development tools.

Appendix F—Using the Visual C++ .NET Debugger—demonstrates key features of the Visual Studio .NET Debugger, which allows a programmer to monitor the execution of applications to locate and remove logic errors. The appendix presents step-by-step instructions, so students learn how to use the debugger in a hands-on manner.

Appendix G—Using the GNU C++ Debugger—demonstrates key features of the GNU C++ Debugger, which allows a programmer to monitor the execution of applications to locate and remove logic errors. The appendix presents step-by-step instructions, so students learn how to use the debugger in a hands-on manner.

Bibliography—lists over 100 books and articles to encourage the student to do further reading on C++ and OOP.

Index—The comprehensive index enables the reader to locate by keyword any term or concept throughout the text.

Software Bundled with *Small C++ How to Program, 5/e*

For the academic educational market only, this textbook is available in a value pack with Microsoft® Visual C++ .NET 2003 Standard Edition integrated development environment as a free supplement. There is no time limit for using the software. [***Note:*** **If you are a professional using this publication, you will have to either purchase the necessary software to build and run the applications in this textbook or download one of the many free compilers available online.**][***Note:*** **If you are a student in a course for which this book is the required textbook, you must purchase your book from your college book-**

store to ensure that you get the value pack with the software. College bookstores will need to order the books directly from Prentice Hall to get the value pack with the software. A caution—used books may not include the software.]

Free C++ Compilers and Trial-Edition C++ Compilers on the Web

Many C++ compilers are available for download from the Web. We discuss several that are available for free or as free-trial versions. Please keep in mind that in many cases, the trial-edition software cannot be used after the (often brief) trial period has expired.

One popular organization that develops free software is the GNU Project (`www.gnu.org`), originally created to develop a free operating system similar to UNIX. GNU offers developer resources, including editors, debuggers and compilers. Many developers use the GCC (GNU Compiler Collection) compilers, available for download from `gcc.gnu.org`. The GCC contains compilers for C, C++, Java and other languages. The GCC compiler is a command-line compiler (i.e., it does not provide a graphical user interface). Many Linux and UNIX systems come with the GCC compiler installed. Red Hat has developed Cygwin (`www.cygwin.com`), an emulator that allows developers to use UNIX commands on Windows. Cygwin includes the GCC compiler.

Borland provides a Windows-based C++ developer product called C++Builder (`www.borland.com/cbuilder/cppcomp/index.html`). The basic C++Builder compiler (a command-line compiler) is free for download. Borland also provides several versions of C++Builder that contain graphical user interfaces (GUIs). These GUIs are formally called integrated development environments (IDEs) and enable the developer to edit, debug and test programs quickly and conveniently. Using an IDE, many of the tasks that involved tedious commands can now be executed via menus and buttons. Some of these products are available on a free-trial basis. For more information on C++Builder, visit

```
www.borland.com/products/downloads/download_cbuilder.html
```

For Linux developers, Borland provides the Borland Kylix development environment. The Borland Kylix Open Edition, which includes an IDE, can be downloaded from

```
www.borland.com/products/downloads/download_kylix.html
```

Borland also provides C++BuilderX—a cross-platform integrated C++ development environment. The free Personal Edition is available from

```
www.borland.com/products/downloads/download_cbuilderx.html
```

The command-line compiler (version 5.6.4) that comes with C++BuilderX was one of several compilers we used to test the programs in this book. Many of the downloads available from Borland require users to register.

The Digital Mars C++ Compiler (`www.digitalmars.com`), is available for Windows and DOS, and includes tutorials and documentation. Readers can download a command-line or IDE version of the compiler. The DJGPP C/C++ development system is available for computers running DOS. DJGPP stands for DJ's GNU Programming Platform, where DJ is for DJ Delorie, the creator of DJGPP. Information on DJGPP can be found at `www.delorie.com/djgpp`. Locations where the compiler can be downloaded at are provided at `www.delorie.com/djgpp/getting.html`.

For a list of other compilers that are available free for download, visit the following sites:

```
www.thefreecountry.com/developercity/ccompilers.shtml
www.compilers.net
```

Warnings and Error Messages on Older C++ Compilers
The programs in this book are designed to be used with compilers that support standard C++. However, there are variations among compilers that may cause occasional warnings or errors. In addition, though the standard specifies various situations that require errors to be generated, it does not specify the messages that compilers should issue. Warnings and error messages vary among compilers—this is normal.

Some older C++ compilers, such as Microsoft Visual C++ 6, Borland C++ 5.5 and various earlier versions of GNU C++, generate error or warning messages in places where newer compilers do not. Although most of the examples in this book will work with these older compilers, there are a few examples that need minor modifications to work with older compilers. The Web site for this book (`www.deitel.com/books/scpphtp5/index.html`) lists the warnings and error messages that are produced by several older compilers and what, if anything, you can do to fix the warnings and errors.

Notes Regarding `using` Declarations and C Standard Library Functions
The C++ Standard Library includes the functions from the C Standard Library. According to the C++ standard document, the contents of the header files that come from the C Standard Library are part of the "`std`" namespace. Some compilers (old and new) generate error messages when `using` declarations are encountered for C functions. We will post a list of these issues at `www.deitel.com/books/scpphtp5/index.html`.

DIVE-INTO™ Series Tutorials for Popular C++ Environments
Our free *Dive-Into™* Series publications, which are available with the resources for *Small C++ How to Program, 5/e* at `www.deitel.com/books/downloads.html`, help students and instructors familiarize themselves with various C++ development tools. These publications include:

- *Dive-Into Microsoft® Visual C++® 6*
- *Dive-Into Microsoft® Visual C++® .NET*
- *Dive-Into Borland™ C++Builder™ Compiler* (command-line version)
- *Dive-Into Borland™ C++Builder™ Personal* (IDE version)
- *Dive-Into GNU C++ on Linux* and *Dive-Into GNU C++ via Cygwin on Windows* (Cygwin is a UNIX emulator for Windows. It includes the GNU C++ compiler)

Each of these tutorials shows how to compile, execute and debug C++ applications in that particular compiler product. Many of these documents also provide step-by-step instructions with screenshots to help readers install the software. Each document overviews the compiler and its online documentation.

Teaching Resources for *Small C++ How to Program, 5/e*

Small C++ How to Program, 5/e, has extensive resources for instructors. The *Instructor's Resource CD (IRCD)* 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 all the code and figures in the text, plus bulleted items that summarize the key points in the text. Instructors can customize the slides. [*Note:* The IRCD is *available only to instructors* through their Prentice Hall representatives. To find your local sales representative, visit

```
vig.prenhall.com/replocator
```

If you need additional help or if you have any questions about the IRCD, please email us at `deitel@deitel.com`. We will respond promptly.]

C++ Multimedia Cyber Classroom, 5/e, Online

C++ How to Program, 5/e and *Small C++ How to Program, 5/e* each include a free, Web-based interactive multimedia ancillary to the book—*The C++ Multimedia Cyber Classroom, 5/e*—available with new books purchased from Prentice Hall for fall 2005 classes. Our Web-based *Cyber Classroom* will include audio walkthroughs of code examples in the text, solutions to about half of the exercises in the book, a free lab manual and more. For more information about the new Web-based *Cyber Classroom*, please visit our Web site at `www.deitel.com` or sign up for the free *Deitel® Buzz Online* e-mail newsletter at

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

Students who use our *Cyber Classrooms* tell us that they like the interactivity and that the *Cyber Classroom* is a powerful reference tool. Professors tell us that their students enjoy using the *Cyber Classroom* and consequently spend more time on the courses, mastering more of the material than in textbook-only courses. For a complete list of our current CD-ROM-based *Cyber Classrooms*, see the *Deitel® Series* page at the beginning of this book, the product listing and ordering information at the end of this book, or visit `www.deitel.com`, `www.prenhall.com/deitel` or `www.InformIT.com/deitel`.

C++ in the Lab

C++ in the Lab: Lab Manual to Accompany C++ How to Program, 5/e, our free online lab manual, complements *C++ How to Program, 5/e*, and *Small C++ How to Program, 5/e*, with hands-on lab assignments designed to reinforce students' comprehension of lecture material. *C++ in the Lab* will be available with new books purchased from Prentice Hall for fall 2005 classes. This lab manual is designed for closed laboratories—regularly scheduled classes supervised by an instructor. Closed laboratories provide an excellent learning environment, because students can use concepts presented in class to solve carefully designed lab problems. Instructors are better able to measure the students' understanding of the material by monitoring the students' progress in lab. This lab manual also can be used for open laboratories, homework and for self-study.

Chapters 1–13 in the lab manual are divided into *Prelab Activities*, *Lab Exercises* and *Postlab Activities*. Each chapter contains objectives that introduce the lab's key topics and an assignment checklist for students to mark which exercises the instructor has assigned.

Solutions to the lab manual's *Prelab Activities*, *Lab Exercises* and *Postlab Activities* are available in electronic form. Instructors can obtain these materials from their regular Prentice Hall representatives; the solutions are not available to students.

Prelab Activities

Prelab Activities are intended to be completed by students after studying each chapter of *Small C++ How to Program, 5/e*. *Prelab Activities* test students' understanding of the textbook material and prepare students for the programming exercises in the lab session. The exercises focus on important terminology and programming concepts and are effective for self-review. *Prelab Activities* include *Matching Exercises*, *Fill-in-the-Blank Exercises*, *Short-Answer Questions*, *Programming-Output Exercises* (determine what short code segments do without actually running the program) and *Correct-the-Code Exercises* (identify and correct all errors in short code segments).

Lab Exercises

The most important section in each chapter is the *Lab Exercises*. These teach students how to apply the material learned in *Small C++ How to Program, 5/e*, and prepare them for writing C++ programs. Each lab contains one or more lab exercises and a debugging problem. The *Lab Exercises* contain the following:

- *Lab Objectives* highlight specific concepts on which the lab exercise focuses.
- *Problem Descriptions* provide the details of the exercise and hints to help students implement the program.
- *Sample Outputs* illustrate the desired program behavior, which further clarifies the problem descriptions and aids the students with writing programs.
- *Program Templates* take complete C++ programs and replace key lines of code with comments describing the missing code.
- *Problem-Solving Tips* highlight key issues that students need to consider when solving the lab exercises.
- *Follow-Up Questions and Activities* ask students to modify solutions to lab exercises, write new programs that are similar to their lab-exercise solutions or explain the implementation choices that were made when solving lab exercises.
- *Debugging Problems* consist of blocks of code that contain syntax errors and/or logic errors. These alert students to the types of errors they are likely to encounter while programming.

Postlab Activities

Professors typically assign *Postlab Activities* to reinforce key concepts or to provide students with more programming experience outside the lab. *Postlab Activities* test the students' understanding of the *Prelab* and *Lab Exercise* material, and ask students to apply their knowledge to creating programs from scratch. The section provides two types of programming activities: coding exercises and programming challenges. Coding exercises are short and serve as review after the *Prelab Activities* and *Lab Exercises* have been completed. The coding exercises ask students to write programs or program segments using key concepts from the textbook. *Programming Challenges* allow students to apply their knowledge to substantial programming exercises. Hints, sample outputs and pseudocode are provided to aid students with these problems. Students who successfully complete the *Programming Challenges* for a chapter have mastered the chapter material. Answers to the programming challenges are available at `www.deitel.com/books/downloads.html`.

CourseCompass^{SM}, WebCT™ and Blackboard™

Selected content from the Deitels' introductory programming language *How to Program* series textbooks, including *Small C++ How to Program, 5/e*, is available to integrate into various popular course management systems, including CourseCompass, Blackboard and WebCT. Course management systems help faculty create, manage and use sophisticated Web-based educational tools and programs. Instructors can save hours of inputting data by using the Deitel course-management-systems content.

Blackboard, CourseCompass and WebCT offer:

- **Features to create and customize an online course**, such as areas to post course information (e.g., policies, syllabi, announcements, assignments, grades, performance evaluations and progress tracking), class and student management tools, a gradebook, reporting tools, page tracking, a calendar and assignments.
- **Communication tools** to help create and maintain interpersonal relationships between students and instructors, including chat rooms, whiteboards, document sharing, bulletin boards and private e-mail.
- **Flexible testing tools** that allow an instructor to create online quizzes and tests from questions directly linked to the text, and that grade and track results effectively. All tests can be inputted into the gradebook for efficient course management. WebCT also allows instructors to administer timed online quizzes.
- **Support materials** for instructors are available in print and online formats.

In addition to the types of tools found in Blackboard and WebCT, CourseCompass from Prentice Hall includes:

- **CourseCompass course home page**, which makes the course as easy to navigate as a book. An expandable table of contents allows instructors to view course content at a glance and to link to any section.
- **Hosting on Prentice Hall's centralized servers**, which allows course administrators to avoid separate licensing fees or server-space issues. Access to Prentice Hall technical support is available.
- **"How Do I" online-support sections** are available for users who need help personalizing course sites, including step-by-step instructions for adding PowerPoint slides, video and more.
- **Instructor Quick Start Guide** helps instructors create online courses using a simple, step-by-step process.

To view free online demonstrations and learn more about these Course Management Systems, which support Deitel content, visit the following Web sites:

- Blackboard: `www.blackboard.com` and `www.prenhall.com/blackboard`
- WebCT: `www.webct.com` and `www.prenhall.com/webct`
- CourseCompass: `www.coursecompass.com` and `www.prenhall.com/coursecompass`

PearsonChoices

Today's students have increasing demands on their time and money, and they need to be resourceful about how, when and where they study. Pearson/Prentice Hall, a division of Pearson Education, has responded to that need by creating PearsonChoices to allow faculty and students to choose from a variety of textbook formats and prices.

Small C++ How to Program, 5/e is our alternative print edition to *C++ How to Program, 5/e. Small C++ How to Program, 5/e* is a smaller text that is focused on Computer Science 1 (CS1) programming courses and is priced lower than our 24-chapter *C++ How to Program, 5/e* and other competing texts in the CS1 market.

***Chapters in Both* Small C++ How to Program, 5/e *and* C++ How to Program, 5/e**
Chapter 1—Introduction to Computers, the Internet and World Wide Web
Chapter 2—Introduction to C++ Programming
Chapter 3—Introduction to Classes and Objects
Chapter 4—Control Statements: Part 1
Chapter 5—Control Statements: Part 2
Chapter 6—Functions and an Introduction to Recursion
Chapter 7—Arrays and Vectors
Chapter 8—Pointers and Pointer-Based Strings
Chapter 9—Classes: A Deeper Look, Part 1
Chapter 10—Classes: A Deeper Look, Part 2
Chapter 11—Operator Overloading: String and Array Objects
Chapter 12—Object-Oriented Programming: Inheritance
Chapter 13—Object-Oriented Programming: Polymorphism

***Appendices in Both* Small C++ How to Program, 5/e *and* C++ How to Program, 5/e**
Operator Precedence and Associativity Chart
ASCII Character Set
Fundamental Types
Number Systems
C++ Internet and Web Resources
Using the Visual C++ .NET Debugger
Using the GNU C++ Debugger

***Chapters in Only* C++ How to Program, 5/e**
Chapter 14—Templates
Chapter 15—Stream Input/Output
Chapter 16—Exception Handling
Chapter 17—File Processing
Chapter 18—Class `string` and String Stream Processing
Chapter 19—Web Programming
Chapter 20—Searching and Sorting
Chapter 21—Data Structures
Chapter 22—Bits, Characters, Streams and Structures
Chapter 23—Standard Template Library
Chapter 24—Other Topics

***Appendices in Only* C++ How to Program, 5/e**
C Legacy-Code Topics
Preprocessor
ATM Case Study Code
UML 2 Diagrams
Introduction to XHTML
XHTML Special Characters

SafariX WebBooks
SafariX Textbooks Online is a new service for college students looking to save money on required or recommended textbooks for academic courses. This secure WebBooks platform creates a new option in the higher education market; an additional choice for students alongside conventional textbooks and online learning services. Pearson provides students with a WebBook at 50% of the cost of its conventional print equivalent.

SafariX WebBooks are viewed through a Web browser connected to the Internet. No special plug-ins are required and no applications need to be downloaded. Students simply log in, purchase access and begin studying. With SafariX Textbooks Online students can search the text, make notes online, print out reading assignments that incorporate their professors' lecture notes and bookmark important passages they want to review later. They can navigate easily to a page number, reading assignment, or chapter. The Table of Contents of each WebBook appears in the left hand column alongside the text.

We are pleased to offer students the *Small C++ How to Program, 5/e* SafariX WebBook available for fall 2005 classes. Visit `www.pearsonchoices.com` for more information. Other Deitel titles available as SafariX WebBooks include *Java How to Program, 6/e*, *Small Java How to Program, 6/e*, *C++ How to Program, 5/e* and *Simply C++: An Application-Driven Tutorial Approach*. Visit `www.safarix.com/tour.html` for more information.

The *Deitel®* *Buzz Online* Free E-mail Newsletter

Our free e-mail newsletter, the *Deitel® Buzz Online* is sent to approximately 38,000 opt-in, registered subscribers 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 our way to notify our readers rapidly about issues related to *Small C++ How to Program, 5/e*. To subscribe, visit

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

Acknowledgments

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

- Andrew B. Goldberg is a graduate of Amherst College, where he earned a bachelor's degree in Computer Science. Andrew updated Chapters 1–13 based on the book's new early-classes presentation and other content revisions.

- Jeff Listfield is a Computer Science graduate of Harvard College. Jeff contributed to Appendices A–C, F and G.
- Cheryl Yaeger graduated from Boston University in three years with a bachelor's degree in Computer Science. Cheryl contributed to Chapters 4, 6, 8, 9 and 13.
- Barbara Deitel, Chief Financial Officer at Deitel & Associates, Inc. researched the quotes at the beginning of each chapter and applied copyedits to the book.
- Abbey Deitel, President of Deitel & Associates, Inc., is an Industrial Management graduate of Carnegie Mellon University. She contributed to the Preface and Chapter 1. She applied copyedits to several chapters in the book, managed the review process and suggested the theme and bug names for the cover of the book.
- Christi Kelsey is a graduate of Purdue University with a bachelor's degree in Management and a minor in Information Systems. Christi contributed to the Preface and Chapter 1. She edited the Index, paged the manuscript and coordinated many aspects of our publishing relationship with Prentice Hall.

We are fortunate to have worked on this project with the talented and dedicated team of publishing professionals at Prentice Hall. We especially appreciate the extraordinary efforts of our Computer Science Editor, Kate Hargett and her boss and our mentor in publishing—Marcia Horton, Editorial Director of Prentice Hall's Engineering and Computer Science Division. Jennifer Cappello did an extraordinary job recruiting the review team and managing the review process from the Prentice Hall side. Vince O'Brien, Tom Manshreck and John Lovell did a marvelous job managing the production of the book. The talents of Paul Belfanti, Carole Anson, Xiaohong Zhu and Geoffrey Cassar are evident in the redesign of the book's interior and the new cover art, and Sarah Parker managed the publication of the book's extensive ancillary package.

We sincerely appreciate the efforts of our fifth-edition reviewers and our fourth-edition post-publication reviewers:

Academic Reviewers
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
Gavin Osborne, Saskatchewan Institute of Applied Science and Technology
Donna Reese, Mississippi State University

Industry Reviewers
Curtis Green, Boeing Integrated Defense Systems
James Huddleston, Independent Consultant
Ed James-Beckham, Borland Software Corporation
Don Kostuch, Independent Consultant

Kriang Lerdsuwanakij, Siemens Limited
William Mike Miller, Edison Design Group, Inc.
Mark Schimmel, Borland International
Vicki Scott, Metrowerks
James Snell, Boeing Integrated Defense Systems
Raymond Stephenson, Microsoft

C++ 4/e Post-Publication Reviewers
Butch Anton, Wi-Tech Consulting
Karen Arlien, Bismarck State College
Jimmy Chen, Salt Lake Community College
Martin Dulberg, North Carolina State University
William Honig, Loyola University
Don Kostuch, Independent Consultant
Earl LaBatt, OPNET Technologies, Inc./ University of New Hampshire
Brian Larson, Modesto Junior College
Kriang Lerdsuwanakij, Siemens Limited
Robert Myers, Florida State University
Gavin Osborne, Saskatchewan Institute of Applied Science and Technology
Wolfgang Pelz, The University of Akron
David Papurt, Independent Consultant
Donna Reese, Mississippi State University
Catherine Wyman, DeVry University, Phoenix
Salih Yurttas, Texas A&M University

Under a tight time schedule, they scrutinized every aspect of the text and made countless suggestions for improving the accuracy and completeness of the presentation.

Well, there you have it! Welcome to the exciting world of 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 will respond promptly, and we will post corrections and clarifications on :

```
www.deitel.com/books/scpphtp5/index.html
```

We hope you enjoy learning with *Small C++ How to Program, Fifth Edition* as much as we enjoyed writing it!

Dr. Harvey M. Deitel
Paul J. Deitel

About the Authors

Dr. Harvey M. Deitel, Chairman and Chief Strategy Officer of Deitel & Associates, Inc., has 43 years experience in the computing field, including extensive industry and academic

experience. Dr. Deitel earned B.S. and M.S. degrees from the Massachusetts Institute of Technology and a Ph.D. from Boston University. He worked on the pioneering virtual-memory operating-systems projects at IBM and MIT that developed techniques now widely implemented in systems such as UNIX, Linux and Windows XP. He has 20 years of college teaching experience, 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. Dr. Deitel has delivered hundreds of professional seminars to major corporations, academic institutions, government organizations and the military. He and Paul are the co-authors of several dozen books and multimedia packages and they are writing many more. With translations published in Japanese, German, Russian, Spanish, Traditional Chinese, Simplified Chinese, Korean, French, Polish, Italian, Portuguese, Greek, Urdu and Turkish, the Deitels' texts have earned international recognition.

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. Through Deitel & Associates, Inc., he has delivered C++, Java, C, Internet and World Wide Web courses to industry clients including IBM, Sun Microsystems, Dell, Lucent Technologies, Fidelity, NASA at the Kennedy Space Center, the National Severe Storm Laboratory, PalmSource, White Sands Missile Range, Rogue Wave Software, Boeing, Stratus, Cambridge Technology Partners, TJX, One Wave, Hyperion Software, Adra Systems, Entergy, CableData Systems and many other organizations. Paul is one of the world's most experienced Java and C++ corporate trainers having taught over 100 professional Java and C++ training courses. He has also lectured on C++ and Java for the Boston Chapter of the Association for Computing Machinery. He and his father, Dr. Harvey M. Deitel, are the world's best-selling Computer Science textbook authors.

About Deitel & Associates, Inc.

Deitel & Associates, Inc., is an internationally recognized corporate training and content-creation organization specializing in computer programming languages, Internet/World Wide Web software technology and object technology education. The company provides instructor-led courses on major programming languages and platforms such as Java, Advanced Java, C, C++, .NET programming languages, XML, Perl, Python; 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 computer companies, government agencies, branches of the military and business organizations. Through its 29-year publishing partnership with Prentice Hall, Deitel & Associates, Inc. publishes leading-edge programming textbooks, professional books, interactive multimedia *Cyber Classrooms*, *Complete Training Courses*, Web-based training courses and course management systems e-content for popular CMSs such as WebCT, Blackboard and Pearson's CourseCompass. 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, see the last few pages of this book or visit:

```
www.deitel.com
```

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

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

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

```
www.deitel.com/books/index.html
```

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