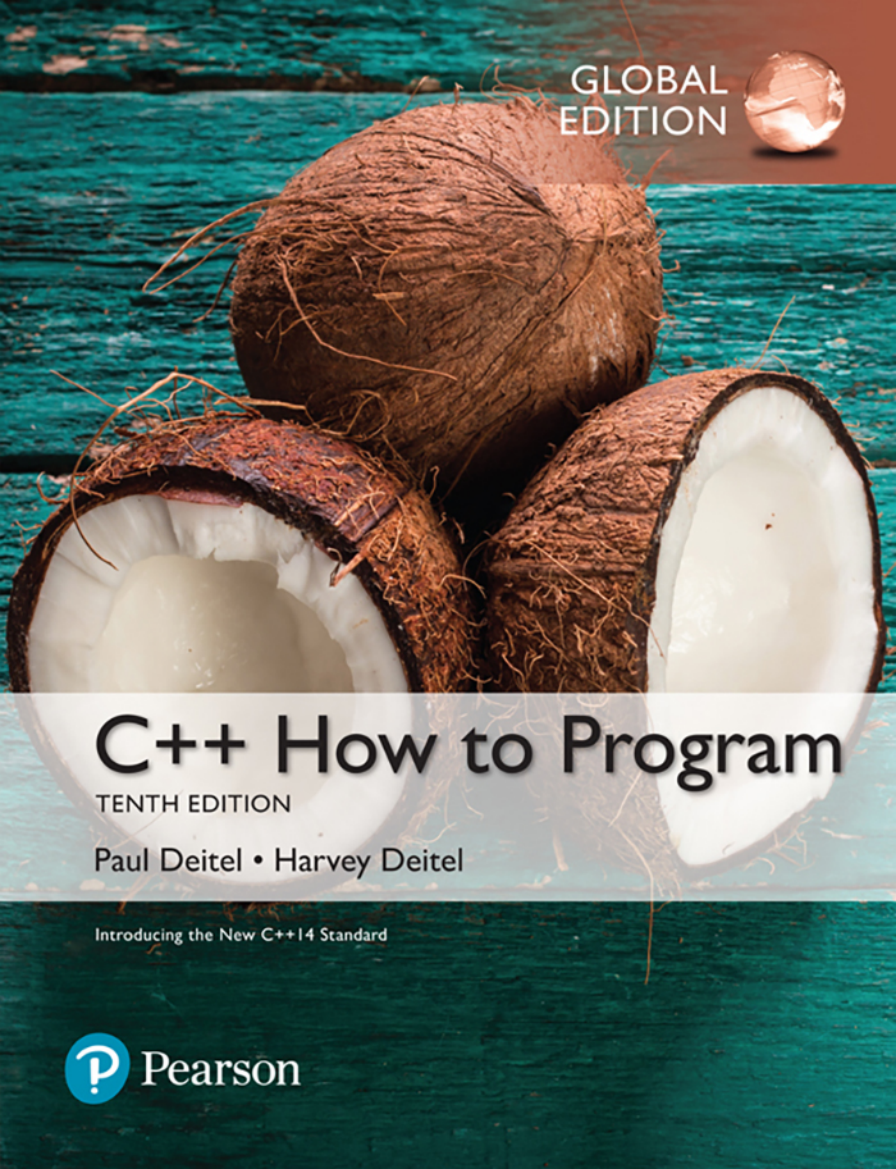


C++ How to Program

TENTH EDITION

Paul Deitel • Harvey Deitel

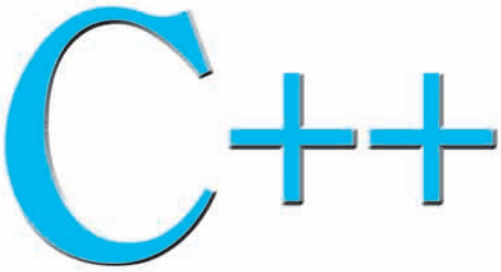
Introducing the New C++14 Standard

Pearson

C++

HOW TO PROGRAM

This page intentionally left blank



HOW TO PROGRAM

Introducing
the New C++14
Standard

TENTH EDITION
GLOBAL EDITION

Paul Deitel
Deitel & Associates, Inc.

Harvey Deitel
Deitel & Associates, Inc.

PEARSON

Boston Columbus Hoboken Indianapolis New York San Francisco
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montreal
Toronto Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

Vice President, Editorial Director: ***Marcia Horton***
Acquisitions Editor: ***Tracy Johnson***
Editorial Assistant: ***Kristy Alaura***
Acquisitions Editor, Global Editions: ***Sourabh Maheshwari***
VP of Marketing: ***Christy Lesko***
Director of Field Marketing: ***Tim Galligan***
Product Marketing Manager: ***Bram Van Kempen***
Field Marketing Manager: ***Demetrius Hall***
Marketing Assistant: ***Jon Bryant***
Director of Product Management: ***Erin Gregg***
Team Lead, Program and Project Management: ***Scott Disanno***
Program Manager: ***Carole Snyder***
Project Manager: ***Robert Engelhardt***
Project Editor, Global Editions: ***K.K. Neelakantan***
Senior Manufacturing Controller, Global Editions: ***Trudy Kimber***
Senior Specialist, Program Planning and Support: ***Maura Zaldivar-Garcia***
Media Production Manager, Global Editions: ***Vikram Kumar***
Cover Art: ***Finevector / Shutterstock***
Cover Design: ***Lumina Datamatics***
R&P Manager: ***Rachel Youdelman***
R&P Project Manager: ***Timothy Nicholls***
Inventory Manager: ***Meredith Maresca***

Credits and acknowledgments borrowed from other sources and reproduced, with permission, in this textbook appear on page 6.

Pearson Education Limited
Edinburgh Gate
Harlow
Essex CM20 2JE
England

and Associated Companies throughout the world

Visit us on the World Wide Web at:
www.pearsonglobaleditions.com

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-15334-2
ISBN 13: 978-1-292-15334-6

Typeset by GEX Publishing Services

Printed and bound in Malaysia

In memory of Marvin Minsky,
a founding father of the
field of artificial intelligence.

It was a privilege to be your student in two graduate courses at M.I.T. Every lecture you gave inspired your students to think beyond limits.

Harvey Deitel

Trademarks

DEITEL and the double-thumbs-up bug are registered trademarks of Deitel and Associates, Inc.

Carnegie Mellon Software Engineering Institute™ is a trademark of Carnegie Mellon University.

CERT® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

UNIX is a registered trademark of The Open Group.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided "as is" without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. Screen shots and icons reprinted with permission from the Microsoft Corporation. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Throughout this book, trademarks are used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state that we are using the names in an editorial fashion only and to the benefit of the trademark owner, with no intention of infringement of the trademark.

Contents

Chapters 23–26 and Appendices F–J are PDF documents posted online at the book's Companion Website, which is accessible from

```
http://www.pearsonglobaleditions.com/deitel
```

See the inside front cover for more information.

Chapters 23–26 and Appendices F–J are PDF documents posted online at the book's Companion Website, which is accessible from

```
http://www.pearsonglobaleditions.com/deitel
```

See the inside front cover for more information.

Preface

Welcome to the C++ computer programming language and *C++ How to Program, Tenth Edition*. We believe that this book and its support materials will give you an informative, challenging and entertaining introduction to C++. The book presents leading-edge computing technologies in a friendly manner appropriate for introductory college course sequences, based on the curriculum recommendations of two key professional organizations—the ACM and the IEEE.[1]

If you haven't already done so, please read the back cover and check out the additional reviewer comments on the inside back cover and the facing page—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 most concepts in the context of complete working programs followed by sample executions, rather than in code snippets. Read the **Before You Begin** section 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

```
http://www.pearsonglobaleditions.com/deitel
```

Use the source code we provide to run each program as you study it.

Contacting the Authors

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

```
http://www.deitel.com/books/cpphtp10
```

Join the Deitel & Associates, Inc. Social Media Communities

Join the Deitel social media communities on

- Facebook®—`http://facebook.com/DeitelFan`
- LinkedIn®—`http://bit.ly/DeitelLinkedIn`

1. *Computer Science Curricula 2013 Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*, December 20, 2013, The Joint Task Force on Computing Curricula, Association for Computing Machinery (ACM), IEEE Computer Society.

- Twitter®—`http://twitter.com/deitel`
- Google+™—`http://google.com/+DeitelFan`
- YouTube®—`http://youtube.com/DeitelTV`

and subscribe to the *Deitel® Buzz Online* newsletter

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

The C++11 and C++14 Standards

These are exciting times in the programming languages community with each of the major languages striving to keep pace with compelling new programming technologies. In the three decades of C++'s development prior to 2011, only a few new versions of the language were released. Now the ISO C++ Standards Committee is committed to releasing a new standard every three years and the compiler vendors are building in the new features promptly. *C++ How to Program, 10/e* is based on the **C++11** and **C++14** standards published in 2011 and 2014, respectively. C++17 is already under active development. Throughout the book, C++11 and C++14 features are marked with the "11" and "14" icons, respectively, that you see here in the margin. Fig. 1 lists the book's first references to the 77 C++11 and C++14 features we discuss.

11
14

C++11 and C++14 features in *C++ How to Program, 10/e*

Chapter 3
In-class initializers

Chapter 4
Keywords new in C++11

Chapter 5
`long long int` type

Chapter 6
Non-deterministic random number generation
Scoped `enum`s
Specifying the type of an `enum`'s constants
Unsigned `long long int`
Using `'` to separate groups of digits in a numeric literals (**C++14**)

Chapter 7
`array` container
`auto` for type inference
List initializing a `vector`
Range-based `for` statement

Chapter 8
`begin`/`end` functions
`nullptr`

Chapter 9
Delegating constructors

Chapter 10
`deleted` member functions
`explicit` conversion operators
List initializing a dynamically allocated array
List initializers in constructor calls
`string` object literals (**C++14**)

Chapter 11
`final` classes
`final` member functions
Inheriting base-class constructors

Chapter 12
`defaulted` member functions
`override` keyword

Chapter 13
`operator bool` for streams

Chapter 14
`quoted` stream manipulator (**C++14**)
`string` objects for file names

Chapter 15
`cbegin`/`cend` container member functions
Compiler fix for `>>` in template types
`crbegin`/`crend` container member functions
`forward_list` container
Global functions `cbegin`/`cend`, `rbegin`/`rend` and `crbegin`/`crend` (**C++14**)
Heterogeneous lookup in associative containers (**C++14**)
Immutable keys in associative containers

Fig. 1 | First references to C++11 and C++14 features in *C++ How to Program, 10/e*. (Part 1 of 2.)

C++11 and C++14 features in *C++ How to Program, 10/e*

Chapter 15 (cont.)
`insert` container member functions return iterators
List initialization of key–value pairs
List initialization of `pairs`
Return value list initialization
`shrink_to_fit` `vector`/`deque` member function

Chapter 16
`all_of` algorithm
`any_of` algorithm
`copy_if` algorithm
`copy_n` algorithm
`equal` algorithm that accepts two ranges (C++14)
`find_if_not` algorithm
Generic lambdas (C++14)
Lambda expressions
`min` and `max` algorithms with `initializer_list` parameters
`minmax` algorithm

Chapter 16 (cont.)
`minmax_element` algorithm
`mismatch` algorithm that accepts two ranges (C++14)
`none_of` algorithm
`random_shuffle` is deprecated (C++14)—replaced with `shuffle` and C++11 random-number generation
`swap` non-member function

Chapter 17
`make_unique` to create a `unique_ptr` (C++14)
`noexcept`
`unique_ptr` smart pointer

Chapter 18
Default type arguments in function templates

Chapter 21
Numeric conversion functions

Chapter 22
Binary literals (C++14)

Chapter 24
Aggregate member initialization (C++14)
`auto` and `decltype(auto)` on return types(C++14)
`constexpr` updated(C++14)
`decltype`
`move` algorithm
Move assignment operators
`move_backward` algorithm
Move constructors
Regular expressions
Rvalue references
`shared_ptr` smart pointer
`static_assert` objects for file names
Trailing return types `for` functions
`tuple` variadic template
`tuple` addressing via type (C++14)
`weak_ptr` smart pointer

Fig. 1 | First references to C++11 and C++14 features in *C++ How to Program, 10/e*. (Part 2 of 2.)

Key Features of C++ How to Program, 10/e

- ***Conforms to the C++11 standard and the new C++14 standard.***
- ***Code thoroughly tested on three popular industrial-strength C++14 compilers.*** We tested the code examples on GNU™ C++ 5.2.1, Microsoft® Visual Studio® 2015 Community edition and Apple® Clang/LLVM in Xcode® 7.
- ***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.
- ***Early coverage of Standard Library containers, iterators and algorithms, enhanced with C++11 and C++14 capabilities.*** The treatment of Standard Library containers, iterators and algorithms in Chapters 15 and 16 has been enhanced with additional C++11 and C++14 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 and C++14 Additional Topics.** This chapter includes discussions of regular expressions, `shared_ptr` and `weak_ptr` smart pointers, move semantics, multithreading, `tuples`, `decltype`, `constexpr` and more (see Fig. 1).

- ***Random-number generation, simulation and game playing.*** To help make programs more secure, we include a treatment of C++11's non-deterministic random-number generation capabilities.
- ***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.*** Several 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.*** On the book's Companion Website we provide Appendix H, Using the Visual Studio Debugger, Appendix I, Using the GNU C++ Debugger and Appendix J, Using the Xcode Debugger.

New in This Edition

- Discussions of the new C++14 capabilities.
- Further integration of C++11 capabilities into the code examples, because the latest compilers are now supporting these features.
- Uniform initialization with list initializer syntax.
- Always using braces in control statements, even for single-statement bodies:

```
if (condition) {
   single-statement or multi-statement body
}
```

- Replaced the `Gradebook` class with `Account`, `Student` and `DollarAmount` class case studies in Chapters 3, 4 and 5, respectively. `DollarAmount` processes monetary amounts precisely for business applications.
- C++14 digit separators in large numeric literals.
- *Type* `&x` is now *Type*`& x` in accordance with industry idiom.
- *Type* `*x` is now *Type*`* x` in accordance with industry idiom.
- Using C++11 scoped `enums` rather than traditional C `enums`.
- We brought our terminology in line with the C++ standard.
- Key terms in summaries now appear in bold for easy reference.
- Removed extra spaces inside `[]`, `()`, `<>` and `{}` delimiters.
- Replaced most `print` member functions with `toString` member functions to make classes more flexible—for example, returning a `string` gives the client code the option of displaying it on the screen, writing it to a file, concatenating it with other `strings`, etc.

- Now using `ostringstream` to create formatted `string`s for items like the `string` representations of a `Time`, rather than outputting formatted data directly to the standard output.
- For simplicity, we deferred using the three-file architecture from Chapter 3 to Chapter 9, so all early class examples define the entire class in a header.
- We reimplement Chapter 10's `Array` class operator-overloading example with `unique_ptr`s in Chapter 24. Using raw pointers and dynamic-memory allocation with `new` and `delete` is a source of subtle programming errors, especially "memory leaks"—`unique_ptr` and the other smart pointer types help prevent such errors.
- Using lambdas rather than function pointers in Chapter 16, Standard Library Algorithms. This will get readers comfortable with lambdas, which can be combined with various Standard Library algorithms to perform functional programming in C++. We're planning a more in-depth treatment of functional programming for *C++ How to Program, 11/e*.
- Enhanced Chapter 24 with additional C++14 features.

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. We worked hard to make this chapter especially accessible to novices. Presenting objects and classes early gets you "thinking about objects" immediately and mastering these concepts more thoroughly.[2]
- ***C++ Standard Library `string`.*** C++ offers *two* types of strings—`string` class objects (which we begin using in Chapter 3) and C-style pointer-based 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 in industry. In new development, you should favor `string` objects.
- ***C++ Standard Library `array`.*** C++ offers *three* types of arrays—`array`s and `vector`s (which we start using in Chapter 7) and C-style, pointer-based arrays which we discuss in Chapter 8. Our primary treatment of arrays uses the Standard Library's `array` and `vector` class templates 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. In new development, you should favor class template `array` and `vector` objects.
- ***Crafting valuable classes.*** A key goal of this book is to prepare you to build valuable reusable classes. 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. In the Chapter 10 case study,

2. For courses that require a late-objects approach, consider our pre-C++11 book *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.

you'll build your own custom `Array` class, then in the Chapter 18 exercises you'll convert it to a class template. You will have truly crafted valuable classes.

- ***Case studies in object-oriented programming.*** We provide several well-engineered real-world case studies, including the `Account` class in Chapter 3, `Student` class in Chapter 4, `DollarAmount` class in Chapter 5, `GradeBook` class in Chapter 7, the `Time` class in Chapter 9, the `Employee` class in Chapters 11–12 and more.
- ***Optional case study: Using the UML to develop an object-oriented design and C++ implementation of an ATM.*** The UML™ (Unified Modeling Language™) is the industry-standard graphical language for modeling object-oriented systems. We introduce the UML in the early chapters. Online Chapters 25 and 26 include an *optional* object-oriented design case study using the UML. We design and fully 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 objects of 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.
- ***Understanding how polymorphism works.*** Chapter 12 contains a detailed diagram and explanation of how C++ typically implements polymorphism, `virtual` functions and dynamic binding "under the hood."
- ***Object-oriented exception handling.*** We integrate basic exception handling early in the book (Chapter 7). 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. 5).
- ***Three programming paradigms.*** We discuss *structured programming*, *object-oriented programming* and *generic programming*.

Hundreds of Code Examples

We include a broad range of example programs selected from computer science, information technology, business, simulation, game playing and other topics. The examples are accessible to students in novice-level and intermediate-level C++ courses (Fig. 2).

Examples	
`Account` class	`BasePlusCommissionEmployee` class
`Array` class case study	Binary tree creation and traversal
`Author` class	`BinarySearch` test program
Bank account program	Card shuffling and dealing
Bar chart printing program	`ClientData` class

Fig. 2 | A sampling of the book's examples. (Part 1 of 2.)

Examples	
`CommissionEmployee` class	Poll analysis program
Comparing `strings`	Polymorphism demonstration
Compilation and linking process	Preincrementing and postincrementing
Compound interest calculations with `for`	`priority_queue` adapter class
Converting `string` objects to C strings	`queue` adapter class
Counter-controlled repetition	Random-access files
Dice game simulation	Random number generation
`DollarAmount` class	Recursive function `factorial`
Credit inquiry program	Rolling a six-sided die 60,000,000 times
`Date` class	`SalariedEmployee` class
Downcasting and runtime type information	`SalesPerson` class
`Employee` class	Searching and sorting algorithms of the Standard Library
`explicit` constructor	Sequential files
`fibonacci` function	`set` class template
`fill` algorithms	`shared_ptr` program
Specializations of function template `printArray`	`stack` adapter class
`generate` algorithms	`Stack` class
`GradeBook` Class	Stack unwinding
Initializing an array in a declaration	Standard Library `string` class program
Input from an `istringstream` object	Stream manipulator `showbase`
Iterative factorial solution	`string` assignment and concatenation
Lambda expressions	`string` member function `substr`
Linked list manipulation	`Student` class
`map` class template	Summing integers with the `for` statement
Mathematical algorithms of the Standard Library	`Time` class
`maximum` function template	`unique_ptr` object managing dynamically allocated memory
Merge sort program	Validating user input with regular expressions
`multiset` class template	`vector` class template
`new` throwing `bad_alloc` on failure	
`PhoneNumber` class	

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

Exercises

- ***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 work on significant social problems. We hope you'll approach these exercises with your own values, politics and beliefs.

Exercises		
Airline Reservations System	Credit Limits	Phishing Scanner
Advanced String-Manipulation	Crossword Puzzle Generator	Pig Latin
Bubble Sort	Cryptograms	Polymorphic Banking Program Using Account Hierarchy
Building Your Own Compiler	De Morgan's Laws	Pythagorean Triples
Building Your Own Computer	Dice Rolling	Salary Calculator
Calculating Salaries	Eight Queens	Sieve of Eratosthenes
`CarbonFootprint` Abstract Class: Polymorphism	Emergency Response	Simple Decryption
Card Shuffling and Dealing	Enforcing Privacy with Cryptography	Simple Encryption
Computer-Assisted Instruction	Facebook User Base Growth	SMS Language
Computer-Assisted Instruction: Difficulty Levels	Fibonacci Series	Spam Scanner
Computer-Assisted Instruction: Monitoring Student Performance	Gas Mileage	Spelling Checker
Computer-Assisted Instruction: Reducing Student Fatigue	Global Warming Facts Quiz	Target-Heart-Rate Calculator
Computer-Assisted Instruction: Varying the Types of Problems	Guess the Number Game	Tax Plan Alternatives; The "Fair Tax"
Cooking with Healthier Ingredients	Hangman Game	Telephone number word generator
Craps Game Modification	Health Records	"The Twelve Days of Christmas" Song
	Knight's Tour	Tortoise and the Hare Simulation
	Limericks	Towers of Hanoi
	Maze Traversal: Generating Mazes Randomly	World Population Growth
	Morse Code	
	Payroll System Modification	
	Peter Minuit Problem	

Fig. 3 | A sampling of the book's exercises.

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).

Drawings and diagrams		
Main text drawings and diagrams		
`Account` class diagrams	`while` repetition statement UML activity diagram	Pass-by-value and pass-by-reference analysis
Data hierarchy	`for` repetition statement UML activity diagram	Inheritance hierarchy diagrams
Multiple-source-file compilation and linking	`do...while` repetition statement UML activity diagram	Function-call stack and activation records
Order in which a second-degree polynomial is evaluated	`switch` multiple-selection statement activity diagram	Recursive calls to function `fibonacci`
`if` single-selection statement activity diagram	C++'s single-entry/single-exit control statements	Pointer arithmetic diagrams
`if...else` double-selection statement activity diagram		`CommunityMember` Inheritance hierarchy

Fig. 4 | A sampling of the book's drawings and diagrams. (Part 1 of 2.)

Drawings and diagrams

Main text drawings and diagrams (cont.)

- `Shape` inheritance hierarchy
- `public`, `protected` and `private` inheritance
- `Employee` hierarchy UML class diagram
- How `virtual` function calls work
- 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

(Optional) 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 `BalanceInquiry` 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 `Car`
- Class diagram for the ATM system model including class `Deposit`
- Activity diagram for a `Deposit` transaction
- Sequence diagram that models a `Deposit` executing

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

Dependency Chart

C++ How to Program, 10/e is appropriate for most introductory one-and-two-course programming sequences, often called CS1 and CS2. The chart in Fig. 5 shows the dependencies among the chapters to help instructors plan their syllabi. The chart shows the book's modular organization.

Teaching Approach

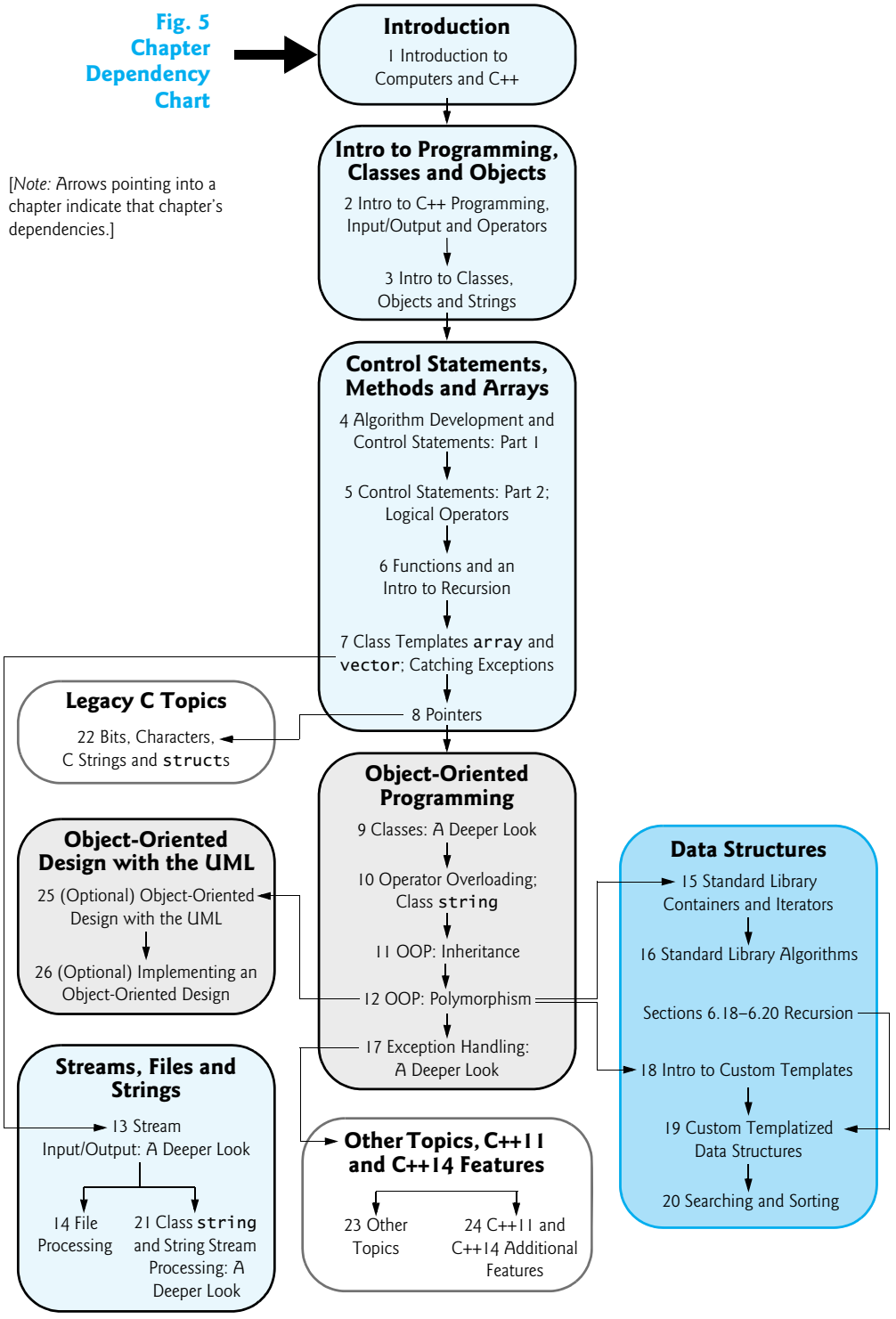
C++ How to Program, 10/e, contains a rich collection of examples. We stress program clarity and concentrate on building well-engineered software.

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.

Rich early coverage of C++ fundamentals. Chapter 2 provides a friendly introduction to C++ programming. We include in Chapters 4 and 5 a clear treatment of control statements and algorithm development.

Fig. 5 Chapter Dependency Chart

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



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 shaded rectangles around the new features in each program.

Using fonts for emphasis. We color the defining occurrence of each key term in **bold colored** 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. We clearly state the 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 eight 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.*

Error-Prevention Tips

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

Performance Tips

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

Portability Tips

These tips help you write code that will run on a variety of platforms.

Software Engineering Observations

These tips highlight architectural and design issues that affect the construction of software systems, especially large-scale systems.

Summary Bullets. We present a section-by-section, bullet-list summary of each chapter. Each key term is in **bold** followed by the page number of the term's defining occurrence.

Index. For convenient reference, we've included an extensive index, with defining occurrences of key terms highlighted with a **bold** page number.

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 which avoid the programming practices that leave systems open to attacks.

We'd like to thank Robert C. Seacord, an adjunct professor in the Carnegie Mellon University School of Computer Science and former Secure Coding Manager at CERT. 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 key guidelines of the *CERT C Secure Coding Standard*.

We've done the same for *C++ How to Program, 10/e*, adhering to key guidelines of the *CERT C++ Secure Coding Standard*, which you can find at:

```
http://www.securecoding.cert.org
```

We were pleased to discover that we've already been recommending many of these coding practices in our books since the early 1990s. 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, 2013).

Online Chapters, Appendices and Other Content

The book's Companion Website, which is accessible at

```
http://www.pearsonglobaleditions.com/deitel
```

(see the inside front cover for your access key) contains the following videos as well as chapters and appendices in searchable PDF format:

- *VideoNotes*—The Companion Website (see the inside front cover for your access key) also includes extensive videos. Watch and listen as co-author Paul Deitel discusses in-depth the key code examples from the book's core programming-fundamentals and object-oriented-programming chapters.
- Chapter 23, Other Topics
- Chapter 24, C++11 and C++14 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
- Building Your Own Compiler exercise descriptions from Chapter 19 (posted at the Companion Website.)

Obtaining the Software Used in *C++ How to Program, 10/e*

We wrote the code examples in *C++ How to Program, 10/e* using the following free C++ development tools:

- Microsoft's free Visual Studio Community 2015 edition, which includes Visual C++ and other Microsoft development tools. This runs on Windows and is available for download at

```
https://www.visualstudio.com/products/visual-studio-community-vs
```

- GNU's free GNU C++ 5.2.1. GNU C++ is already installed on most Linux systems and can also be installed on Mac OS X and Windows systems. There are many versions of Linux—known as Linux distributions—that use different techniques for performing software upgrades. Check your distribution's online documentation for information on how to upgrade GNU C++ to the latest version. GNU C++ is available at

```
http://gcc.gnu.org/install/binaries.html
```

- Apple's free Xcode, which OS X users can download from the Mac App Store—click the app's icon in the dock at the bottom of your screen, then search for Xcode in the app store.

Instructor Supplements

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

- ***Solutions Manual*** contains solutions to most of the end-of-chapter exercises. We include *Making a Difference* exercises, many 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.

  Solutions are *not* provided for "project" exercises. Check out our Programming Projects Resource Center for lots of additional exercise and project possibilities.

```
http://www.deitel.com/ProgrammingProjects
```

- ***Test Item File*** of multiple-choice questions.
- ***Customizable PowerPoint® slides*** containing all the code and figures in the text, plus bulleted items that summarize 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.

An optional 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

```
http://www.myprogramminglab.com
```

Acknowledgments

We'd like to thank Barbara Deitel of Deitel & Associates, Inc. for long hours devoted to this project. She painstakingly researched the new capabilities of C++11 and C++14.

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. Kristy Alaura 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.

Finally, thanks to Abbey Deitel, former President of Deitel & Associates, Inc., and a graduate of Carnegie Mellon University's Tepper School of Management where she received a B.S. in Industrial Management. Abbey managed the business operations of Deitel & Associates, Inc. for 17 years, along the way co-authoring a number of our publications, including the previous *C++ How to Program* editions' versions of Chapter 1.

Reviewers

We wish to acknowledge the efforts of our reviewers. Over its ten editions, the book has been scrutinized by academics teaching C++ courses, current and former members of the C++ standards committee and industry experts using C++ to build industrial-strength, high-performance systems. They provided countless suggestions for improving the presentation. Any remaining flaws in the book are our own.

Tenth Edition reviewers: Chris Aburime, Minnesota State Colleges and Universities System; Gašper Ažman, A9.com Search Technologies and Co-Author of *C++ Today: The Beast is Back*; Danny Kalev, Intel and Former Member of the C++ Standards Committee; Renato Golin, LLVM Tech Lead at Linaro and Co-Owner for the ARM Target in LLVM; Gordon Hogenson, Microsoft, Author of *Foundations of C++/CLI: The Visual C++ Language for .NET 3*; Jonathan Wakely, Redhat, ISO C++ Committee Secretary; José Antonio González Seco, Parliament of Andalusia; Dean Michael Berris, Google, Maintainer of cpp-netlib and Former ISO C++ Committee Member.

Ninth Edition post-publication academic reviewers: Stefano Basagni, Northeastern University; Amr Elkady, Diablo Valley College; Chris Aburime, Minnesota State Colleges and Universities System.

Other recent edition reviewers: Virginia Bailey (Jackson State University), Ed James-Beckham (Borland), 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), Linda M. Krause (Elmhurst College), Wing-Ning Li (University of Arkansas), Dean Mathias (Utah State University), Robert A. McLain (Tidewater Community College), James P. McNellis (Microsoft Corporation), Robert Myers (Florida State University), Gavin Osborne (Saskatchewan Institute of Applied Science and Technology), Amar Raheja (California State Polytechnic University, Pomona), April Reagan (Microsoft), Robert C. Seacord (Secure Coding Manager at SEI/CERT, author of *Secure Coding in C and C++*), 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:

```
deitel@deitel.com
```

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

Paul Deitel
Harvey Deitel

Acknowledgments for the Global Edition

Pearson would like to thank and acknowledge the following people for their contributions to the Global Edition.

Contributors
Rosanne Els, University of Kawazulu Natal; Almasi S. Maguya, Mzumbe University; Khyat Sharma.

Reviewers
T.V. Gopal, Anna University; Subrajeet Mohapatra, Birla Institute of Technology; Prabhat Verma, Harcourt Butler Technological Institute.

About the Authors

Paul Deitel, CEO and Chief Technical Officer of Deitel & Associates, Inc., has over 30 years of experience in computing. He is a graduate of MIT, where he studied Information Technology. He holds the Java Certified Programmer and Java Certified Developer designations and is an Oracle Java Champion. Paul was also named as a Microsoft® Most Valuable Professional (MVP) for C# in 2012–2014. Through Deitel & Associates, Inc., he has delivered hundreds of programming courses worldwide to 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, Nortel Networks, Puma, iRobot, Invensys and many more. He and his co-author, Dr. Harvey 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 over 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 studied computing in each of these programs before they spun off Computer Science programs. 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. The Deitels' publications have earned international recognition, with translations published in Japanese, German, Russian, Spanish, French, Polish, Italian, Simplified Chinese, Traditional Chinese, Korean, Portuguese, Greek, Urdu and Turkish. Dr. Deitel has delivered hundreds of programming courses to academic, corporate, government and military clients.

About 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, Internet and web software technology, and Android and iOS app development. The company's clients include academic institutions, many of the world's largest corporations, government agencies and branches of the military. The company offers instructor-led training courses delivered at client sites worldwide on major programming languages and platforms, including C++, C, Java™, Android app development, iOS app development, Swift™, Visual C#®, Visual Basic®, Internet and web programming and a growing list of additional programming and software-development courses.

Before You Begin

This section contains information you should review before using this book and instructions to ensure that your computer is set up properly to compile the example programs.

Font and Naming Conventions

We use fonts to distinguish between features, such as menu names, menu items, and other elements that appear in your IDE (Integrated Development Environment), such as Microsoft's Visual Studio. Our convention is to emphasize IDE features in a sans-serif bold **Helvetica** font (for example, **File** menu) and to emphasize program text in a sans-serif `Lucida` font (for example, `bool x = true;`).

Obtaining the Software Used in *C++ How to Program, 10/e*

Before reading this book, you should download and install a C++ compiler. We wrote *C++ How to Program, 10/e*'s code examples using the following free C++ development tools:

- Microsoft's free Visual Studio Community 2015 edition, which includes the Visual C++ compiler and other Microsoft development tools. This runs on Windows and is available for download at

```
https://www.visualstudio.com/products/visual-studio-community-vs
```

- GNU's free GNU C++ 5.2.1 compiler. GNU C++ is already installed on most Linux systems and also can be installed on Mac OS X and Windows systems. There are many versions of Linux, known as Linux distributions, that use different techniques for performing software upgrades. Check your distribution's online documentation for information on how to upgrade GNU C++ to the latest version. GNU C++ is available at

```
http://gcc.gnu.org/install/binaries.html
```

- Apple's free Xcode, which OS X users can download from the Mac App Store—click the app's icon in the dock at the bottom of your Mac screen, then search for Xcode in the app store.

We also provide links to our getting-started videos for each of these C++ tools at:

```
http://www.deitel.com/books/cpphtp10
```

Obtaining the Code Examples

The examples for *C++ How to Program, 10/e* are available for download at

```
http://www.pearsonglobaleditions.com/deitel
```

Click the **Download Code Examples** link to download the ZIP archive file to your computer. Write down the location where you saved the file—most browsers will save the file into your user account's `Downloads` folder.

Throughout the book, steps that require you to access our example code on your computer assume that you've extracted the examples from the ZIP file and placed them in `C:\examples` on Windows or in your user account's `Documents` directory on other platforms. You can extract them anywhere you like, but if you choose a different location, you'll need to update our steps accordingly.

Creating Projects

In Section 1.10, we demonstrate how to compile and run programs with

- Microsoft Visual Studio Community 2015 edition on Windows (Section 1.10.1)
- GNU C++ 5.2.1 on Linux (Section 1.10.2)
- Apple Xcode on OS X (Section 1.10.3)

For GNU C++ and Xcode, you must compile your programs with C++14. To do so in GNU C++, include the option `-std=c++14` when you compile your code, as in:

```
g++ -std=c++14 YourFileName.cpp
```

For Xcode, after following Section 1.10.3's steps to create a project:

1. Select the root node at the top of the Xcode **Project** navigator.
2. Click the **Build Settings** tab in the **Editors** area.
3. Scroll down to the **Apple LLVM 7.0 - Language - C++** section.
4. For the **C++ Language Dialect** option, select **C++14 [–std=c++14]**.

Getting Your C++ Questions Answered

As you read the book, if you have questions, we're easy to reach at

```
deitel@deitel.com
```

We'll respond promptly.

In addition, the web is loaded with programming information. An invaluable resource for nonprogrammers and programmers alike is the website

```
http://stackoverflow.com
```

on which you can:

- Search for answers to most common programming questions
- Search for error messages to see what causes them
- Ask programming questions to get answers from programmers worldwide
- Gain valuable insights about programming in general

Online C++ Documentation

For documentation on the C++ Standard Library, visit

```
http://cppreference.com
```

and be sure to check out the C++ FAQ at

```
https://isocpp.org/faq
```

Introduction to Computers and C++

1

Objectives

In this chapter you'll learn:

- Exciting recent developments in the computer field.
- Computer hardware, software and networking basics.
- The data hierarchy.
- The different types of programming languages.
- Basic object-technology concepts.
- Some basics of the Internet and the World Wide Web.
- A typical C++ program development environment.
- To test-drive a C++ application.
- Some key recent software technologies.
- How computers can help you make a difference.

Outline

1.1 Introduction

Welcome to C++—a powerful computer programming language that's appropriate for technically oriented people with little or no programming experience, and for experienced programmers to use in building substantial information systems. You're already familiar with the powerful tasks computers perform. Using this textbook, you'll write instructions commanding computers to perform those kinds of tasks. *Software* (i.e., the instructions you write) controls *hardware* (i.e., computers).

You'll learn *object-oriented programming*—today's key programming methodology. You'll create many *software objects* that model *things* in the real world.

C++ is one of today's most popular software development languages. This text provides an introduction to programming in C++11 and C++14—the latest versions standardized through the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC).

11
14

As of 2008 there were more than a billion general-purpose computers in use. Today, various websites say that number is approximately two billion, and according to the real-time tracker at `gsmaintelligence.com`, there are now more mobile devices than there are people in the world. According to the International Data Corporation (IDC), the number of mobile Internet users will top two billion in 2016.[1] Smartphone sales surpassed personal computer sales in 2011.[2] Tablet sales were expected to overtake personal-computer sales

1. `https://www.idc.com/getdoc.jsp?containerId=prUS40855515`.
2. `http://www.mashable.com/2012/02/03/smartphone-sales-overtake-pcs/`.

by 2015.[3] By 2017, the smartphone/tablet app market is expected to exceed $77 billion.[4] This explosive growth is creating significant opportunities for programming mobile applications.

1.2 Computers and the Internet in Industry and Research

These are exciting times in the computer field. Many of the most influential and successful businesses of the last two decades are technology companies, including Apple, IBM, Hewlett Packard, Dell, Intel, Motorola, Cisco, Microsoft, Google, Amazon, Facebook, Twitter, eBay and many more. These companies are major employers of people who study computer science, computer engineering, information systems or related disciplines. At the time of this writing, Apple was the most valuable company in the world. Figure 1.1 provides a few examples of the ways in which computers are improving people's lives in research, industry and society.

Name	Description
Electronic health records	These might include a patient's medical history, prescriptions, immunizations, lab results, allergies, insurance information and more. Making this information available to health care providers across a secure network improves patient care, reduces the probability of error and increases overall efficiency of the health-care system, helping control costs.
Human Genome Project	The Human Genome Project was founded to identify and analyze the 20,000+ genes in human DNA. The project used computer programs to analyze complex genetic data, determine the sequences of the billions of chemical base pairs that make up human DNA and store the information in databases which have been made available over the Internet to researchers in many fields.
AMBER™ Alert	The AMBER (America's Missing: Broadcast Emergency Response) Alert System is used to find abducted children. Law enforcement notifies TV and radio broadcasters and state transportation officials, who then broadcast alerts on TV, radio, computerized highway signs, the Internet and wireless devices. AMBER Alert recently partnered with Facebook, whose users can "Like" AMBER Alert pages by location to receive alerts in their news feeds.
World Community Grid	People worldwide can donate their unused computer processing power by installing a free secure software program that allows the World Community Grid (`http://www.worldcommunitygrid.org`) to harness unused capacity. This computing power, accessed over the Internet, is used in place of expensive supercomputers to conduct scientific research projects that are making a difference—providing clean water to third-world countries, fighting cancer, growing more nutritious rice for regions fighting hunger and more.

Fig. 1.1 | A few uses for computers. (Part 1 of 3.)

3. `http://www.forbes.com/sites/louiscolumbus/2014/07/18/gartner-forecasts-tablet-shipments-will-overtake-pcs-in-2015/`.
4. `http://www.entrepreneur.com/article/236832`.

Name	Description
Cloud computing	**Cloud computing** allows you to use software, hardware and information stored in the "cloud"—i.e., accessed on remote computers via the Internet and available on demand—rather than having it stored on your personal computer. These services allow you to increase or decrease resources to meet your needs at any given time, so they can be more cost effective than purchasing expensive hardware to ensure that you have enough storage and processing power to meet your needs at their peak levels. Using cloud-computing services shifts the burden of managing these applications from the business to the service provider, saving businesses money.
Medical imaging	X-ray computed tomography (CT) scans, also called CAT (computerized axial tomography) scans, take X-rays of the body from hundreds of different angles. Computers are used to adjust the intensity of the X-rays, optimizing the scan for each type of tissue, then to combine all of the information to create a 3D image. MRI scanners use a technique called magnetic resonance imaging, also to produce internal images noninvasively.
GPS	Global Positioning System (GPS) devices use a network of satellites to retrieve location-based information. Multiple satellites send time-stamped signals to the GPS device, which calculates the distance to each satellite, based on the time the signal left the satellite and the time the signal arrived. This information helps determine the device's exact location. GPS devices can provide step-by-step directions and help you locate nearby businesses (restaurants, gas stations, etc.) and points of interest. GPS is used in numerous location-based Internet services such as check-in apps to help you find your friends (e.g., Foursquare and Facebook), exercise apps such as RunKeeper that track the time, distance and average speed of your outdoor jog, dating apps that help you find a match nearby and apps that dynamically update changing traffic conditions.
Robots	Robots can be used for day-to-day tasks (e.g., iRobot's Roomba vacuuming robot), entertainment (e.g., robotic pets), military combat, deep sea and space exploration (e.g., NASA's Mars rover Curiosity) and more. RoboEarth (`www.roboearth.org`) is "a World Wide Web for robots." It allows robots to learn from each other by sharing information and thus improving their abilities to perform tasks, navigate, recognize objects and more.
E-mail, Instant Messaging, Video Chat and FTP	Internet-based servers support all of your online messaging. E-mail messages go through a mail server that also stores the messages. Instant Messaging (IM) and Video Chat apps, such as Facebook Messenger, AIM, Skype, Yahoo! Messenger, Google Hangouts, Trillian and others allow you to communicate with others in real time by sending your messages and live video through servers. FTP (file transfer protocol) allows you to exchange files between multiple computers (for example, a client computer such as your desktop and a file server) over the Internet.
Internet TV	Internet TV set-top boxes (such as Apple TV, Android TV, Roku and TiVo) allow you to access an enormous amount of content on demand, such as games, news, movies, television shows and more, and they help ensure that the content is streamed to your TV smoothly.

Fig. 1.1 | A few uses for computers. (Part 2 of 3.)

Name	Description
Streaming music services	Streaming music services (such as Apple Music, Pandora, Spotify, Last.fm and more) allow you listen to large catalogues of music over the web, create customized "radio stations" and discover new music based on your feedback.
Game programming	Global video-game revenues are expected to reach $107 billion by 2017 (`http://www.polygon.com/2015/4/22/8471789/worldwide-video-games-market-value-2015`). The most sophisticated games can cost over $100 million to develop, with the most expensive costing half a billion dollars (`http://www.gamespot.com/gallery/20-of-the-most-expensive-games-ever-made/2900-104/`). Bethesda's *Fallout 4* earned $750 million in its first day of sales (`http://fortune.com/2015/11/16/fallout4-is-quiet-best-seller/`)!

Fig. 1.1 | A few uses for computers. (Part 3 of 3.)

1.3 Hardware and Software

Computers can perform calculations and make logical decisions phenomenally faster than human beings can. Many of today's personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime. *Supercomputers* are already performing *thousands of trillions (quadrillions)* of instructions per second! China's National University of Defense Technology's Tianhe-2 supercomputer can perform over 33 quadrillion calculations per second (33.86 *petaflops*)![5] To put that in perspective, *the Tianhe-2 supercomputer can perform in one second about 3 million calculations for every person on the planet!* And supercomputing "upper limits" are growing quickly.

Computers process data under the control of sequences of instructions called **computer programs**. These programs guide the computer through ordered actions specified by people called computer **programmers**. The programs that run on a computer are referred to as **software**. In this book, you'll learn a key programming methodology that's enhancing programmer productivity, thereby reducing software development costs—*object-oriented programming*.

A computer consists of various devices referred to as hardware (e.g., the keyboard, screen, mouse, hard disks, memory, DVD drives and processing units). Computing costs are *dropping dramatically*, owing to rapid developments in hardware and software technologies. Computers that might have filled large rooms and cost millions of dollars decades ago are now inscribed on silicon chips smaller than a fingernail, costing perhaps a few dollars each. Ironically, silicon is one of the most abundant materials on Earth—it's an ingredient in common sand. Silicon-chip technology has made computing so economical that computers have become a commodity.

1.3.1 Moore's Law

Every year, you probably expect to pay at least a little more for most products and services. The opposite has been the case in the computer and communications fields, especially with regard to the hardware supporting these technologies. For many decades, hardware costs have fallen rapidly.

5. `http://www.top500.org`.

Every year or two, the capacities of computers have approximately *doubled* inexpensively. This remarkable trend often is called **Moore's Law**, named for the person who identified it in the 1960s, Gordon Moore, co-founder of Intel—a leading manufacturer of the processors in today's computers and embedded systems. Moore's Law and related observations apply especially to the amount of memory that computers have for programs, the amount of secondary storage (such as disk storage) they have to hold programs and data over longer periods of time, and their processor speeds—the speeds at which they *execute* their programs (i.e., do their work). These increases make computers more capable, which puts greater demands on programming-language designers to innovate.

Similar growth has occurred in the communications field—costs have plummeted as enormous demand for communications *bandwidth* (i.e., information-carrying capacity) has attracted intense competition. We know of no other fields in which technology improves so quickly and costs fall so rapidly. Such phenomenal improvement is truly fostering the *Information Revolution*.

1.3.2 Computer Organization

Regardless of differences in *physical* appearance, computers can be envisioned as divided into various **logical units** or sections (Fig. 1.2).

Logical unit	Description
Input unit	This "receiving" section obtains information (data and computer programs) from **input devices** and places it at the disposal of the other units for processing. Most user input is entered into computers through keyboards, touch screens and mouse devices. Other forms of input include receiving voice commands, scanning images and barcodes, reading from secondary storage devices (such as hard drives, DVD drives, Blu-ray Disc™ drives and USB flash drives—also called "thumb drives" or "memory sticks"), receiving video from a webcam and having your computer receive information from the Internet (such as when you stream videos from YouTube® or download e-books from Amazon). Newer forms of input include position data from a GPS device, and motion and orientation information from an *accelerometer* (a device that responds to up/down, left/right and forward/backward acceleration) in a smartphone or game controller (such as Microsoft® Kinect® for Xbox®, Wii™ Remote and Sony® PlayStation® Move).
Output unit	This "shipping" section takes information the computer has processed and places it on various **output devices** to make it available for use outside the computer. Most information that's output from computers today is displayed on screens (including touch screens), printed on paper ("going green" discourages this), played as audio or video on PCs and media players (such as Apple's iPods) and giant screens in sports stadiums, transmitted over the Internet or used to control other devices, such as robots and "intelligent" appliances. Information is also commonly output to secondary storage devices, such as hard drives, DVD drives and USB flash drives. Popular recent forms of output are smartphone and game controller vibration, and virtual reality devices like Oculus Rift.

Fig. 1.2 | Logical units of a computer. (Part 1 of 2.)

Logical unit	Description
Memory unit	This rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed. The memory unit also retains processed information until it can be placed on output devices by the output unit. Information in the memory unit is *volatile*—it's typically lost when the computer's power is turned off. The memory unit is often called either **memory**, **primary memory** or **RAM** (Random Access Memory). Main memories on desktop and notebook computers contain as much as 128 GB of RAM, though 2 to 16 GB is most common. GB stands for gigabytes; a gigabyte is approximately one billion bytes. A **byte** is eight bits. A bit is either a `0` or a `1`.
Arithmetic and logic unit (ALU)	This "manufacturing" section performs *calculations*, such as addition, subtraction, multiplication and division. It also contains the *decision* mechanisms that allow the computer, for example, to compare two items from the memory unit to determine whether they're equal. In today's systems, the ALU is implemented as part of the next logical unit, the CPU.
Central processing unit (CPU)	This "administrative" section coordinates and supervises the operation of the other sections. The CPU tells the input unit when information should be read into the memory unit, tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices. Many of today's computers have multiple CPUs and, hence, can perform many operations simultaneously. A **multi-core processor** implements multiple processors on a single integrated-circuit chip—a *dual-core processor* has two CPUs, a *quad-core processor* has four and an *octa-core processor* has eight. Today's desktop computers have processors that can execute billions of instructions per second. To take full advantage of multi-core architecture you need to write multithreaded applications, which we introduce in Section 24.3.
Secondary storage unit	This is the long-term, high-capacity "warehousing" section. Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your *hard drive*) until they're again needed, possibly hours, days, months or even years later. Information on secondary storage devices is *persistent*—it's preserved even when the computer's power is turned off. Secondary storage information takes much longer to access than information in primary memory, but its cost per unit is much less. Examples of secondary storage devices include hard drives, DVD drives and USB flash drives, some of which can hold over 2 TB (TB stands for terabytes; a terabyte is approximately one trillion bytes). Typical hard drives on desktop and notebook computers hold up to 2 TB, and some desktop hard drives can hold up to 6 TB.

Fig. 1.2 | Logical units of a computer. (Part 2 of 2.)

1.4 Data Hierarchy

Data items processed by computers form a **data hierarchy** that becomes larger and more complex in structure as we progress from the simplest data items (called "bits") to richer ones, such as characters and fields. Figure 1.3 illustrates a portion of the data hierarchy.

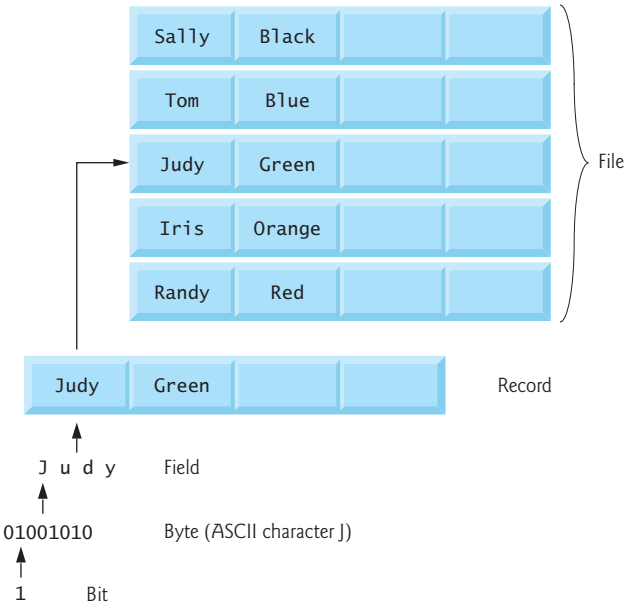


Fig. 1.3 | Data hierarchy.

Bits

The smallest data item in a computer can assume the value `0` or the value `1`. It's called a **bit** (short for "binary digit"—a digit that can assume one of *two* values). Remarkably, the impressive functions performed by computers involve only the simplest manipulations of `0`s and `1`s—*examining a bit's value*, *setting a bit's value* and *reversing a bit's value* (from `1` to `0` or from `0` to `1`).

Characters

It's tedious for people to work with data in the low-level form of bits. Instead, they prefer to work with *decimal digits* (0–9), *letters* (A–Z and a–z), and *special symbols* (e.g., $, @, %, &, *, (,), –, +, ", :, ? and /). Digits, letters and special symbols are known as **characters**. The computer's **character set** is the set of all the characters used to write programs and represent data items. Computers process only `1`s and `0`s, so a computer's character set represents every character as a pattern of `1`s and `0`s. C++ supports various character sets (including **Unicode**®), with some requiring more than one byte per character. Unicode supports many of the world's languages. See Appendix B for more information on the **ASCII (American Standard Code for Information Interchange)** character set—the popular subset of Unicode that represents uppercase and lowercase letters, digits and some common special characters.

Fields

Just as characters are composed of bits, **fields** are composed of characters or bytes. A field is a group of characters or bytes that conveys meaning. For example, a field consisting of uppercase and lowercase letters can be used to represent a person's name, and a field consisting of decimal digits could represent a person's age.

Records

Several related fields can be used to compose a **record**. In a payroll system, for example, the record for an employee might consist of the following fields (possible types for these fields are shown in parentheses):

- Employee identification number (a whole number)
- Name (a string of characters)
- Address (a string of characters)
- Hourly pay rate (a number with a decimal point)
- Year-to-date earnings (a number with a decimal point)
- Amount of taxes withheld (a number with a decimal point).

Thus, a record is a group of related fields. In the preceding example, all the fields belong to the *same* employee. A company might have many employees and a payroll record for each.

Files

A **file** is a group of related records. [*Note:* More generally, a file contains arbitrary data in arbitrary formats. In some operating systems, a file is viewed simply as a *sequence of bytes*—any organization of the bytes in a file, such as organizing the data into records, is a view created by the application programmer.] It's not unusual for an organization to have many files, some containing billions, or even trillions, of characters of information.

Database

A **database** is a collection of data organized for easy access and manipulation. The most popular model is the *relational database*, in which data is stored in simple *tables*. A table includes *records* and *fields*. For example, a table of students might include first name, last name, major, year, student ID number and grade-point-average fields. The data for each student is a record, and the individual pieces of information in each record are the fields. You can *search*, *sort* and otherwise manipulate the data based on its relationship to multiple tables or databases. For example, a university might use data from the student database in combination with data from databases of courses, on-campus housing, meal plans, etc.

Big Data

The amount of data being produced worldwide is enormous and growing quickly. According to IBM, approximately 2.5 quintillion bytes (2.5 *exabytes*) of data are created daily[6] and according to Salesforce.com, 90% of the world's data was created in just the past 12 months![7] According to an IDC study, the global data supply will reach 40 *zettabytes* (equal to 40 trillion gigabytes) annually by 2020.[8] Figure 1.4 shows some common byte measurements. **Big data** applications deal with massive amounts of data and this field is growing quickly, creating lots of opportunity for software developers. According to a study by Gartner Group, over 4 million IT jobs globally were expected to support big data in 2015.[9]

6. http://www-01.ibm.com/software/data/bigdata/what-is-big-data.html.
7. https://www.salesforce.com/blog/2015/10/salesforce-channel-ifttt.html.
8. http://recode.net/2014/01/10/stuffed-why-data-storage-is-hot-again-really/.
9. http://tech.fortune.cnn.com/2013/09/04/big-data-employment-boom/.