



Pearson International Edition

*Introduction to
Programming*

with **C++**

COMPREHENSIVE VERSION

Y. Daniel Liang

THE FUNDAMENTALS FIRST APPROACH

INTRODUCTION TO
PROGRAMMING
WITH

C++

COMPREHENSIVE VERSION

Y. Daniel Liang

Armstrong Atlantic State University

E1067519



Pearson Education International

If you purchased this book within the United States or Canada you should be aware that it has been wrongfully imported without the approval of the Publisher or the Author.

Vice President and Editorial Director, ECS: *Marcia J. Horton*

Executive Editor: *Tracy Dunkelberger*

Assistant Editor: *Carole Snyder*

Editorial Assistant: *Christianna Lee*

Executive Managing Editor: *Vince O'Brien*

Managing Editor: *Camille Trentacoste*

Production Editor: *Rose Kernan*

Director of Creative Services: *Paul Belfanti*

Creative Director: *Juan Lopez*

Cover Designer: *Heather Scott*

Managing Editor, AV Management and Production: *Patricia Burns*

Art Editor: *Xiaohong Zhu*

Director, Image Resource Center: *Melinda Reo*

Manager, Rights and Permissions: *Zina Arabia*

Manager, Visual Research: *Beth Brenzel*

Manager, Cover Visual Research and Permissions: *Karen Sanatar*

Manufacturing Manager, ESM: *Alexis Heydt-Long*

Manufacturing Buyer: *Lisa McDowell*

Executive Marketing Manager: *Robin O'Brien*

Marketing Assistant: *Mack Patterson*

Photo Credits:

Cover: Two-headed serpent pectoral, 1400-1520 AD, Aztec. Bridgeman Art Library.

Part Opener: The Goddess Chalchihuitlicue, found in the Valley of Mexico, 1300-1500 AD (stone), Aztec / Musée de l'Homme, Paris, France / Bridgeman Art Library.

Chapter Opener: Mayan God Shel, Mexico. Photographer: Philip Coblenz. Courtesy Brand X Pictures.



© 2007 Pearson Education, Inc.

Pearson Prentice Hall

Pearson Education, Inc.

Upper Saddle River, NJ 07458

All rights reserved. No part of this book may be reproduced in any form or by any means, without permission in writing from the publisher.

Pearson Prentice Hall™ is a trademark of Pearson Education, Inc.

All other trademarks or product names are the property of their respective owners.

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of the theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages in connection with, or arising out of, the furnishing, performance, or use of these programs.

Printed in the United States of America

10 9 8 7 6 5 4 3 2

ISBN: 0-13-159418-4

Pearson Education LTD.

Pearson Education Australia PTY Ltd.

Pearson Education Singapore, Pte. Ltd.

Pearson Education North Asia Ltd.

Pearson Education Canada, Inc.

Pearson Educación de Mexico, S.A. de C.V.

Pearson Education—Japan

Pearson Education Malaysia, Pte. Ltd.

Pearson Education, Inc., Upper Saddle River, New Jersey

To Samantha, Michael, and Michelle

PREFACE

After ten years of Java momentum, C++ remains a popular programming language widely used in the industry and taught in academia. Java is ideal for developing GUI, Internet and cross-platform applications, whereas C++ excels in system programming such as operating systems and compilers. Java and C++ will co-exist and compliment each other.

There are many C++ texts. What distinguishes this book from others are the *fundamentals-first* approach and the writing style. The *fundamentals-first approach* introduces fundamental programming concepts on control statements, loops, functions, and arrays before introducing object-oriented programming. The writing style of this book can be summarized in two words: *clear* and *concise*. The concepts are *clearly* explained using simple, short, and stimulating examples. The explanations are *concisely* presented with many figures and tables.

fundamentals-first

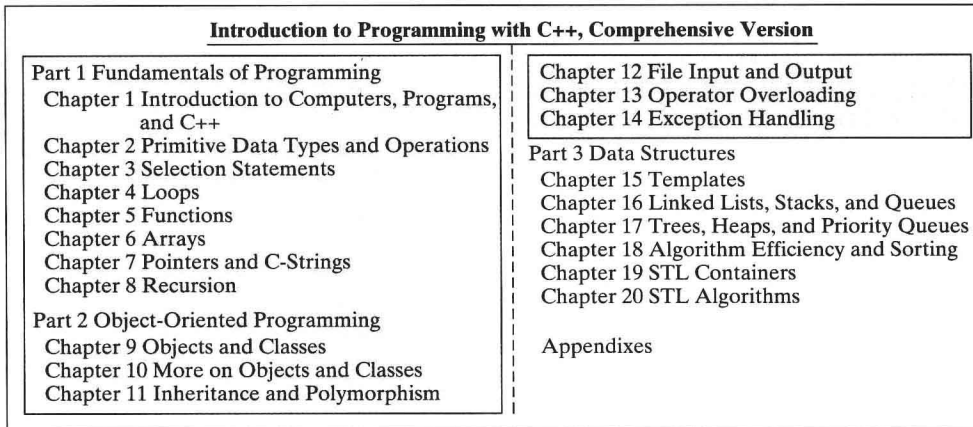
clear
concise

Versions

The book is available in two versions:

- The Brief Version (Chapters 1–14).
- The Comprehensive Version (Chapters 1–20).

The following diagram summarizes the contents in the comprehensive version:



The *Brief Version* introduces fundamentals of programming, problem-solving, and object-oriented programming. This version is suitable for a course on introduction to problem solving and object-oriented programming.

brief version

The *Comprehensive Version* contains all the chapters in the brief version. Additionally, it covers data structures and advanced C++ programming.

comprehensive version

Teaching Strategies

There are several strategies in teaching C++. This book adopts the *fundamentals-first* strategy, proceeding at a steady pace through all the necessary and important basic concepts, then moving to object-oriented programming, and then to the use of the object-oriented approach to build interesting applications with exception handling, I/O, and data structures.

fundamentals-first

fundamental programming techniques

From my own experience, confirmed by the experiences of many colleagues, we have found that learning basic logic and *fundamental programming techniques* like loops and step-wise refinement is essential for new programmers to succeed. Students who cannot write code in procedural programming are not able to learn object-oriented programming. A good introduction on primitive data types, control statements, functions, and arrays prepares students to learn object-oriented programming.

using OOP effectively

The fundamentals-first approach reinforces object-oriented programming (OOP) by first presenting the procedural solutions and then demonstrating how they can be improved using the object-oriented approach. Students can learn when and how to apply OOP effectively.

object-early failed?

At every SIGCSE (Computer Science Education) conference prior to 2005, the object-early approach was trumpeted and the voice for the fundamentals-first approach was muted. This has been changed when some former proponents of object-early began to air their frustrations and declared that object-early failed. This book is fundamentals-first and *object-right*. OOP is introduced just right in time after fundamental programming techniques are covered.

object-right

problem solving

Programming isn't just syntax, classes, or objects. It is really *problem solving*. Loops, functions, and arrays are fundamental techniques for problem solving. From fundamental programming techniques to object-oriented programming, there are many layers of abstraction. Classes are simply a layer of abstraction. Applying the concept of abstraction in the design and implementation of software projects is the key to developing software. The overriding objective of this book, therefore, is to teach students to use many layers of abstraction in solving problems and to see problems in small detail and in large scale. The examples and exercises throughout this book center on problem solving and foster the concept of developing reusable components and using them to create practical projects.

Learning Strategies

practice

A programming course is quite different from other courses. In a programming course, you learn from examples, from practice, and from mistakes. You need to devote a lot of time to writing programs, testing them, and fixing errors.

programmatic solution

For first-time programmers, learning C++ is like learning any high-level programming language. The fundamental point in learning programming is to develop the critical skills of formulating programmatic solutions for real problems and translating them into programs using selection statements, loops, and functions.

object-oriented programming

Once you acquire the basic skills of writing programs using loops, functions, and arrays, you can begin to learn object-oriented programming. You will learn how to develop object-oriented software using class encapsulation and class inheritance.


Pedagogical Features

teaching by example

learning by doing

The philosophy of the Liang Series is *teaching by example and learning by doing*. Basic features are explained by example so that you can learn by doing. This book uses the following elements to get the most from the material:

- **Objectives** list what students should have learned from the chapter. This will help them to determine whether they have met the objectives after completing the chapter.
- **Introduction** opens the discussion with a brief overview of what to expect from the chapter.
- **Examples**, carefully chosen and presented in an easy-to-follow style, teach programming concepts. This book uses many small, simple, and stimulating examples to demonstrate important ideas.
- **Chapter Summary** reviews the important subjects that students should understand and remember. It helps them to reinforce the key concepts they have learned in the chapter.

- **Optional Sections** cover nonessential but valuable features. Instructors may choose to include or skip an optional section or to cover it later. The section headers of optional sections are marked by  .
- **Review Questions** are grouped by sections to help students track their progress and evaluate their learning.
- **Programming Exercises** are grouped by sections to provide students with opportunities to apply on their own the new skills they have learned. The level of difficulty is rated easy (no asterisk), moderate (*), hard (**), or challenging (***) . The trick of learning programming is practice, practice, and practice. To that end, this book provides a great many exercises.
- **Interactive Self-Test** lets students test their knowledge interactively online. The Self-Test is accessible from the Companion Website. It provides more than one thousand multiple-choice questions organized by sections in each chapter. The Instructor Resource Website contains the quiz generator with additional multiple-choice questions.
- **Notes, Tips, and Cautions** are inserted throughout the text to offer valuable advice and insight on important aspects of program development.



Note

Provides additional information on the subject and reinforces important concepts.



Tip

Teaches good programming style and practice.

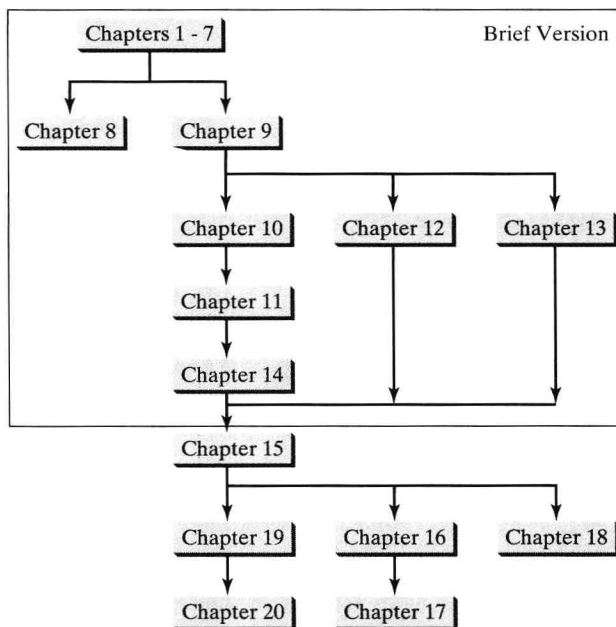


Caution

Helps students steer away from the pitfalls of programming errors.

Chapter Dependency

The following diagram shows the chapter dependency. Note that Chapter 8, “Recursion,” Chapter 12, “File Input and Output,” and Chapter 13, “Operator Overloading,” can be covered in flexible orders.



C++ Development Tools

You can use a text editor, such as the Windows Notepad or WordPad, to create C++ programs, and you can compile and run the programs from the command window. You also can use a C++ development tool, such as Visual C++, Dev-C++, and C++Builder. These tools support an integrated development environment (IDE) for rapidly developing C++ programs. Editing, compiling, building, executing, and debugging programs are integrated in one graphical user interface. Using these tools effectively will greatly increase your programming productivity. How to create, compile, and run programs using Visual C++ and Dev-C++ is introduced in Chapter 1. Detailed tutorials on Visual C++ and C++Builder are in the supplements on the Companion Website.

The programs in this book have been tested on Visual C++, C++Builder, and the GNU C++ compiler.

Companion Website

The companion Website at www.prenhall.com/liang or www.cs.armstrong.edu/liang/cpp contains the following resources:

- Answers to review questions
- Solutions to even-numbered programming exercises
- Source code for the examples in the book
- Interactive Self-Test (organized by sections for each chapter)
- Supplements
- Resource links
- Errata

Supplements

The text covers the essential subjects. The supplements extend the text to introduce additional topics that might be of interest to readers. The following supplements are available from the Companion Website.

Supplements for Introduction to Programming with C++	
Part I General Supplements A Glossary B Installing and Configuring C++ Compiler C Compiling and Running C++ from the Command Window D C++ Coding Style Guidelines	Part III Preprocessor A Preprocessor Directives Part IV Advanced C++ Topics A Multiple Inheritance B Namespaces C Operator Keywords Part V Legacy Topics A Redirecting Input/Output B Using Command-Line Argument C C goto Statements D C printf Statements
Part II IDE Supplements A Visual C++ 2005 Tutorial B Learning C++ Effectively with Visual C++ C Dev-C++ Tutorial D C++Builder Tutorial E Learning C++ Effectively with C++Builder	

Instructor Resource Website

The Instructor Resource Website accessible from www.prenhall.com/liang contains the following resources:

- Microsoft PowerPoint slides with interactive buttons to view full-color, syntax-highlighted source code and to run programs without leaving the slides.

- Sample exams. In general, each exam has four parts:
 1. Multiple-choice questions or short-answer questions (most of these are different from the questions in the self-test on the Companion Website)
 2. Correct programming errors
 3. Trace programs
 4. Write programs
- Solutions to all the exercises. Students will have access to the solutions of even-numbered exercises in the book's Companion Website.
- Web-based quiz generator. (Instructors can choose chapters to generate quizzes from a large database of more than 2000 questions.)
- Online quiz. (Students can take the online quiz for each chapter, and a quiz report will be sent to the instructor.)

Some readers have requested the materials from the Instructor Resource Website. Please understand that these are for instructors only. Such requests will not be answered.

Acknowledgments

I would like to thank Ray Greenlaw and my colleagues at Armstrong Atlantic State University for enabling me to teach what I write and for supporting me in writing what I teach. I thank the students in my C++ class for proofreading the draft.

This book was greatly enhanced thanks to the following reviewers:

Dan Lipsa	Armstrong Atlantic State University
Hui Liu	Missouri State University
Ronald Marsh	University of North Dakota
Charles Nelson	Rock Valley College
Martha Sanchez	University of Texas at Dallas
Kate Stewart	Tallahassee Community College
Margaret Tseng	Montgomery College
Barbara Tulley	Elizabethtown College

It is a great pleasure, honor, and privilege to work with Prentice Hall. I would like to thank Marcia Horton, Tracy Dunkelberger, Robin O'Brien, Christianna Lee, Carole Snyder, Mack Patterson, Vince O'Brien, Camille Trentacoste, Xiaohong Zhu, Rose Kernan and their colleagues for organizing, producing, and promoting this project, as well as Shelly Gerger-Knechtel for copy editing and Scott Disanno for proofreading.

As always, I am indebted to my wife, Samantha, for her love, support, and encouragement.

Y. Daniel Liang
liang@armstrong.edu
www.cs.armstrong.edu/liang/cpp

CONTENTS

PART 1 ■ FUNDAMENTALS OF PROGRAMMING	19
Chapter 1 Introduction to Computers, Programs, and C++	21
1.1 Introduction	22
1.2 What Is a Computer?	22
1.3 Programs	25
1.4 Operating Systems	27
1.5 (Optional) Number Systems	28
1.6 History of C++	31
1.7 A Simple C++ Program	32
1.8 C++ Program Development Cycle	33
1.9 Developing C++ Programs Using Visual C++	35
1.10 Developing C++ Programs Using Dev-C++	41
1.11 Developing C++ Programs from Command Line on Windows	46
1.12 Developing C++ Programs on UNIX	47
Chapter 2 Primitive Data Types and Operations	51
2.1 Introduction	52
2.2 Writing Simple Programs	52
2.3 Reading Input from the Keyboard	54
2.4 Omitting the <code>std::</code> Prefix	55
2.5 Identifiers	56
2.6 Variables	56
2.7 Assignment Statements and Assignment Expressions	57
2.8 Named Constants	59
2.9 Numeric Data Types and Operations	60
2.10 Numeric Type Conversions	66
2.11 Character Data Type and Operations	68
2.12 Case Studies	70
2.13 Programming Style and Documentation	75
2.14 Programming Errors	76
2.15 Debugging	77

Chapter 3	Selection Statements	85
3.1	Introduction	86
3.2	The <code>bool</code> Data Type	86
3.3	<code>if</code> Statements	87
3.4	Example: Guessing Birth Dates	89
3.5	Logical Operators	91
3.6	<code>if ... else</code> Statements	94
3.7	Nested <code>if</code> Statements	95
3.8	Example: Computing Taxes	97
3.9	Example: A Math Learning Tool	100
3.10	<code>switch</code> Statements	101
3.11	Conditional Expressions	104
3.12	Formatting Output	104
3.13	Operator Precedence and Associativity	107
3.14	Enumerated Types	109
Chapter 4	Loops	119
4.1	Introduction	120
4.2	The <code>while</code> Loop	120
4.3	The <code>do-while</code> Loop	124
4.4	The <code>for</code> Loop	126
4.5	Which Loop to Use?	129
4.6	Nested Loops	130
4.7	Case Studies	131
4.8	(Optional) Keywords <code>break</code> and <code>continue</code>	136
4.9	Example: Displaying Prime Numbers	138
4.10	(Optional) Simple File Input and Output	140
Chapter 5	Functions	153
5.1	Introduction	154
5.2	Creating a Function	154
5.3	Calling a Function	155
5.4	<code>void</code> Functions	157
5.5	Passing Parameters by Values	159
5.6	Passing Parameters by References	161
5.7	Overloading Functions	163
5.8	Function Prototypes	165
5.9	Default Arguments	167

5.10	Case Study: Computing Taxes with Functions	168
5.11	Reusing Functions by Different Programs	170
5.12	Case Study: Generating Random Characters	171
5.13	The Scope of Variables	173
5.14	The Math Functions	177
5.15	Function Abstraction and Stepwise Refinement	177
5.16	(Optional) Inline Functions	185
Chapter 6 Arrays		199
6.1	Introduction	200
6.2	Array Basics	200
6.3	Passing Arrays to Functions	207
6.4	Returning Arrays from Functions	210
6.5	Searching Arrays	212
6.6	Sorting Arrays	215
6.7	Two-Dimensional Arrays	218
6.8	(Optional) Multidimensional Arrays	225
Chapter 7 Pointers and C-Strings		237
7.1	Introduction	238
7.2	Pointer Basics	238
7.3	Passing Arguments by References with Pointers	241
7.4	Arrays and Pointers	242
7.5	Using <code>const</code> with Pointers	244
7.6	Returning Pointers from Functions	245
7.7	Dynamic Memory Allocation	247
7.8	Case Studies: Counting the Occurrences of Each Letter	249
7.9	Characters and Strings	252
7.10	Case Studies: Checking Palindromes	260
Chapter 8 Recursion		269
8.1	Introduction	270
8.2	Example: Factorials	270
8.3	Example: Fibonacci Numbers	272
8.4	Problem Solving Using Recursion	274
8.5	Recursive Helper Functions	276
8.6	Towers of Hanoi	279
8.7	Recursion versus Iteration	282

PART 2 ■ OBJECT-ORIENTED PROGRAMMING	289
Chapter 9 Objects and Classes	291
9.1 Introduction	292
9.2 Defining Classes for Objects	292
9.3 Constructors	294
9.4 Object Names	294
9.5 Separating Declaration from Implementation	298
9.6 Accessing Object Members via Pointers	300
9.7 Creating Dynamic Objects on Heap	301
9.8 The C++ <code>string</code> Class	301
9.9 Data Field Encapsulation	305
9.10 The Scope of Variables	308
9.11 The <code>this</code> Pointer	310
9.12 Passing Objects to Functions	311
9.13 Array of Objects	313
9.14 Class Abstraction and Encapsulation	315
9.15 Case Study: The Loan Class	315
9.16 Constructor Initializer Lists	319
Chapter 10 More on Objects and Classes	329
10.1 Introduction	330
10.2 Immutable Objects and Classes	330
10.3 Preventing Multiple Declarations	332
10.4 Instance and Static Members	334
10.5 Destructors	337
10.6 Copy Constructors	339
10.7 Customizing Copy Constructors	342
10.8 <code>friend</code> Functions and <code>friend</code> Classes	344
10.9 Object Composition	346
10.10 Case Study: The Course Class	347
10.11 Case Study: The <code>StackOfIntegers</code> Class	350
10.12 The C++ <code>vector</code> Class	353
Chapter 11 Inheritance and Polymorphism	361
11.1 Introduction	362
11.2 Base Classes and Derived Classes	362

11.3	Generic Programming	368
11.4	Constructors and Destructors	368
11.5	Redefining Functions	371
11.6	Polymorphism and Virtual Functions	372
11.7	The protected Keyword	375
11.8	Abstract Classes and Pure Virtual Functions	376
11.9	Dynamic Casting	380
Chapter 12 File Input and Output		391
12.1	Introduction	392
12.2	Text I/O	392
12.3	Formatting Output	396
12.4	Member Functions: getLine, get, and put	397
12.5	fstream and File Open Modes	400
12.6	Testing Stream States	401
12.7	Binary I/O	403
12.8	Random Access File	410
12.9	Updating Files	413
Chapter 13 Operator Overloading		417
13.1	Introduction	418
13.2	The Rational Class	418
13.3	Operator Functions	423
13.4	Overloading the Shorthand Operators	425
13.5	Overloading the [] Operators	425
13.6	Overloading the Unary Operators	427
13.7	Overloading the ++ and -- Operators	427
13.8	Overloading the << and >> Operators	428
13.9	Object Conversion	430
13.10	The New Rational Class	431
13.11	Overloading the = Operators	438
Chapter 14 Exception Handling		443
14.1	Introduction	444
14.2	Exception-Handling Overview	444
14.3	Exception-Handling Advantages	446
14.4	Exception Classes	447
14.5	Custom Exception Classes	450
14.6	Multiple Catches	455