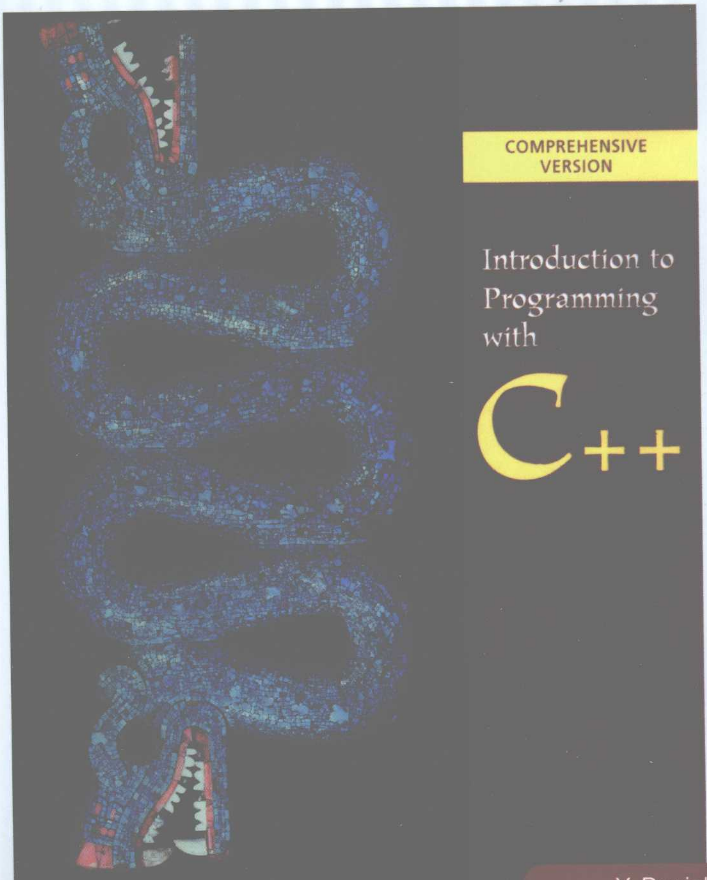
HZ BOOKS
华章教育

经 典 原 版 书 库

PRENTI
HALL

C++程序设计

(英文版)



附赠光盘

(美) Y. Daniel Liang 著
阿姆斯特朗亚特兰大州立大学

机械工业出版社
China Machine Press

经 典 原 版 书 库

C++程序设计

（英文版）

Introduction to Programming with C++

Comprehensive Version

（美） Y. Daniel Liang 著
阿姆斯特朗亚特兰大州立大学

机 械 工 业 出 版 社
China Machine Press

本书版权登记号：图字：01-2007-1830

图书在版编目（CIP）数据

C++程序设计（英文版）/（美）梁（Liang, Y. D.）著. —北京：机械工业出版社，2008.5

书名原文：Introduction to Programming with C++: Comprehensive Version
（经典原版书库）

ISBN 978-7-111-23996-3

Ⅰ. C…　Ⅱ. 梁…　Ⅲ. C语言－程序设计－英文　Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第056363号

机械工业出版社（北京市西城区百万庄大街22号　邮政编码　100037）
责任编辑：迟振春
北京京北制版厂印刷 · 新华书店北京发行所发行
2008年5月第1版第1次印刷
145mm × 210mm · 20.75印张
标准书号：ISBN 978-7-111-23996-3
　　　　　ISBN 978-7-89482-647-3（光盘）
定价：45.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换
本社购书热线：（010）68326294

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.

fundamentals-first

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.

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:

Introduction to Programming with C++, Comprehensive Version

brief 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.

comprehensive version

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

Teaching Strategies

fundamentals-first

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.

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?
object-right
problem solving

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.

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.

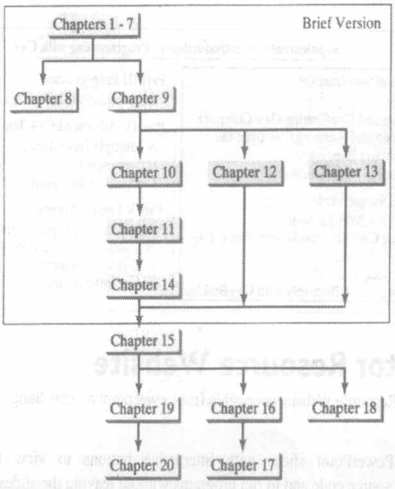
Note
Provides additional information on the subject and reinforces important concepts.

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 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

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

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-Knechtl 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

BRIEF CONTENTS

CONTENTS