教 育 部 高 等 教 育 司 推 荐
国外优秀信息科学与技术系列教学用书

C++程序设计

（第二版　影印版）

PROGRAMMING IN C++

(Second Edition)

■ Nell Dale
Chip Weems
Mark Headington

高 等 教 育 出 版 社
Higher Education Press
Jones and Bartlett Publishers

教 育 部 高 等 教 育 司 推 荐
国外优秀信息科学与技术系列教学用书

C++程序设计

（第二版 影印版）

PROGRAMMING IN C++

(Second Edition)

Nell Dale
Chip Weems
Mark Headington

高 等 教 育 出 版 社
Jones and Bartlett Publishers

图字：01-2001-1035 号

Programming in C++ , 2nd ed.
Nell Dale, Chip Weems & Mark Headington

Original English Language Edition Published By
Jones and Bartlett Publishers, Inc.
One Exert Plaza
Boston, MA 02116

图书在版编目(CIP)数据

C+ +程序设计：英文 /（美）戴尔（Dale,N.）等. －2 版.
—北京：高等教育出版社，2001(2003 重印)
ISBN 7－04－010038－X

Ⅰ.C…　Ⅱ.戴…　Ⅲ.C 语言－程序设计－英文　Ⅳ.
TP312

中国版本图书馆 CIP 数据核字(2001)第 19765 号

C+ +程序设计（第二版）
Nell Dale　等

出版发行	高等教育出版社	**购书热线**	010－64054588
社　　址	北京市东城区沙滩后街 55 号	**免费咨询**	800－810－0598
邮政编码	100009	**网　　址**	http://www.hep.edu.cn
传　　真	010－64014048		http://www.hep.com.cn
经　　销	新华书店北京发行所		
印　　刷	北京外文印刷厂		
开　　本	787×1092　1/16	**版　　次**	2001 年 5 月第 1 版
印　　张	48.75	**印　　次**	2003 年 5 月第 3 次印刷
字　　数	1 085 000	**定　　价**	41.00 元

前　　言

20 世纪末，以计算机和通信技术为代表的信息科学和技术，对世界的经济、军事、科技、教育、文化、卫生等方面的发展产生了深刻的影响，由此而兴起的信息产业已经成为世界经济发展的支柱。进入 21 世纪，各国为了加快本国的信息产业，加大了资金投入和政策扶持。

为了加快我国信息产业的进程，在我国《国民经济和社会发展第十个五年计划纲要》中，明确提出“以信息化带动工业化，发挥后发优势，实现社会生产力的跨越式发展。”信息产业的国际竞争将日趋激烈。在我国加入 WTO 后，我国信息产业将面临国外竞争对手的严峻挑战。竞争成败最终将取决于信息科学和技术人才的多少与优劣。

在 20 世纪末，我国信息产业虽然得到迅猛发展，但与国际先进国家相比，差距还很大。为了赶上并超过国际先进水平，我国必须加快信息技术人才的培养，特别要培养一大批具有国际竞争能力的高水平的信息技术人才，促进我国信息产业和国家信息化水平的全面提高。为此，教育部高等教育司根据教育部吕福源副部长的意见，在长期重视推动高等学校信息科学和技术的教学的基础上，将实施超前发展战略，采取一些重要举措，加快推动高等学校的信息科学和技术等相关专业的教学工作。在大力宣传、推荐我国专家编著的面向 21 世纪和“九五”重点的信息科学和技术课程教材的基础上，在有条件的高等学校的某些信息科学和技术课程中推动使用国外优秀教材的影印版进行英语或双语教学，以缩短我国在计算机教学上与国际先进水平的差距，同时也有助于强化我国大学生的英语水平。

为了达到上述目的，在分析一些出版社已影印相关教材，一些学校已试用影印教材进行教学的基础上，教育部高等教育司组织并委托高等教育出版社开展国外优秀信息科学和技术优秀教材及其教学辅助材料的引进研究与影印出版的试点工作。为推动用影印版教材进行教学创造条件。

本次引进的系列教材的影印出版工作，是在对我国高校的信息科学和技术专业的课程与美国高校的进行对比分析的基础上展开的；所影印出版的教材均由我国主要高

校的信息科学和技术专家组成的专家组，从国外近两年出版的大量最新教材中精心筛选评审通过的内容新、有影响的优秀教材；影印教材的定价原则上应与我国大学教材价格相当。

教育部高等教育司将此影印系列教材推荐给高等学校，希望有关教师选用，使用后有什么意见和建议请及时反馈。也希望有条件的出版社，根据影印教材的要求，积极参加此项工作，以便引进更多、更新、更好的外国教材和教学辅助材料。

同时，感谢国外有关出版公司对此项引进工作的配合，欢迎更多的国外公司关心并参与此项工作。

教育部高等教育司

二〇〇一年四月

*To you, and to all of our students for whom it has begun
and without whom it would never have been completed.*

N.D.   C.W.   M.H.

preface

The first edition of *Programming in C++* was prepared in response to requests for a straightforward, no-frills introduction to C++. Although this second edition incorporates numerous changes, including reorganization of chapter material, one thing has not changed: our commitment to the student. As always, our efforts are directed toward making the sometimes difficult concepts of computer science more accessible to all students.

This edition of *Programming in C++* continues to reflect our experience that topics once considered too advanced can be taught in the first course. For example, preconditions and postconditions are used in the context of the algorithm walk-through, in the development of testing strategies, and as interface documentation for user-written functions. Data abstraction and abstract data types (ADTs) are explained in conjunction with the C++ class mechanism, forming a natural lead-in to object-oriented programming.

Changes in the Second Edition

The second edition incorporates the following changes:

- *Conformance to ISO/ANSI standard C++.* ISO/ANSI standard C++ (officially approved in July 1998) is used throughout the book, including relevant portions of the new C++ standard library. However, readers with pre-standard C++ compilers are also supported. A new appendix discusses how to modify the textbook's programs to compile and run successfully with an earlier compiler.
- *An earlier introduction to classes, data abstraction, and object-oriented concepts.* Chapters 11–16 of the first edition have been reorganized into the following Chapters 11–15:
  11 Structured Types, Data Abstraction, and Classes
  12 Arrays
  13 Array-Based Lists
  14 Object-Oriented Software Development
  15 Recursion

The visible changes are the deletion of two chapters ("Records" and "Multidimensional Arrays"), whose contents have been merged into Chapters 11 and 12, respectively, and the movement of material on classes and data abstraction (covered in Chapter 15 of the first edition) to Chapter 11. With this reorganization, the concept of the C++ class as both a structuring mechanism and a tool for abstraction now comes earlier in the book.

Introducing classes before arrays has several benefits. In their first exposure to composite types, many students find it easier to comprehend accessing a component by name rather than by position. Chapter 12 on arrays can now rather easily introduce the idea of an array of class objects or an array of structs. Also, Chapter 13, which deals with the list as an ADT, can now be handled in a better way, namely, encapsulating both the data representation (an array) and the length variable within a class, rather than the first edition's approach of using two loosely coupled variables (an array and a separate length variable) to represent the list. Finally, with three chapters' worth of exposure to classes and objects, students reading Chapter 14 can focus on the more difficult aspects of the chapter: inheritance, composition, and dynamic binding.

A natural result of this reorganization is that the chapter "Object-Oriented Software Development" comes earlier in the sequence: Chapter 14 rather than the first edition's Chapter 15.

C++ and Object-Oriented Programming

Some educators reject the C++ language as too permissive and too conducive to writing cryptic, unreadable programs. Our experience does not support this view, *provided that the use of language features is modeled appropriately.* We have found that with careful instruction in software engineering and a programming style that is straightforward, disciplined, and free of intricate language features, students can learn to use C++ to produce clear, readable code.

It must be emphasized that although we use C++ as a vehicle for teaching computer science concepts, the book is not a language manual and does not attempt to cover all of C++. Certain language features—templates, exceptions, operator overloading, default arguments, and mechanisms for advanced forms of inheritance, to name a few—are omitted in an effort not to overwhelm the beginning student with too much too fast.

There are diverse opinions about when to introduce the topic of object-oriented programming (OOP). Some educators advocate an immersion in OOP from the very beginning, whereas others (for whom this book is intended) favor a more heterogeneous approach in which both functional decomposition and object-oriented design are presented as design tools. The chapter organization of *Programming in C++* reflects a transitional approach to OOP. Although we provide an early preview of object-oriented design in Chapter 4, we delay a focused discussion until Chapter 14. The sequence of topics in Chapters 1 through 13 mirrors our belief that OOP is best understood after a firm grounding in algorithm design, control abstraction, and data abstraction with classes.

Features

Web Links Special Web icons found throughout the book prompt students to visit the text's companion Web site located at `www.jbpub.com/dale` for additional information about selected topics. These Web Links give students instant access to real-world applications of material presented in the text. The Web Links are updated on a regular basis to ensure that students receive the most recent information available on the Internet.

Goals Each chapter begins with a list of learning objectives for the student. These goals are reinforced and tested in the end-of-chapter exercises.

Programming Examples Included in most chapters, programming examples present a problem and discuss its solution. We then code the solution in C++. We also show sample test data and output and follow up with a discussion of program testing.

Testing and Debugging These sections consider the implications of the chapter material with regard to testing of programs. They conclude with a list of testing and debugging hints.

Quick Checks These questions test the student's recall of major points associated with the chapter goals. Upon reading each question, the student immediately should know the answer, which he or she can then verify by glancing at the answers at the end of the section. The page number on which the concept is discussed appears at the end of each question so that the student can review the material in the event of an incorrect response.

Exam Preparation Exercises To help the student prepare for tests, these questions usually have objective answers and are designed to be answerable with a few minutes of work. Answers to selected questions are given in the back of the book, and the remaining questions are answered in the *Instructor's Guide*.

Programming Warm-up Exercises These questions provide the student with experience in writing C++ code fragments. The student can practice the syntactic constructs in each chapter without the burden of writing a complete program.

Programming Problems These exercises require the student to design solutions and write complete programs.

Supplements

Instructor's Guide and Test Bank The *Instructor's Guide* features chapter-by-chapter teaching notes, answers to the balance of the exercises, and a compilation of exam questions with answers. The *Instructor's Guide* is available to adopters on request from Jones and Bartlett.

Instructor's ToolKit CD-ROM Also available to adopters upon request from the publisher is a powerful teaching tool entitled "Instructor's ToolKit." This CD-ROM contains an electronic version of the *Instructor's Guide*, a computerized test bank, PowerPoint lecture presentations, and the complete programs from the text (see below).

Programs The programs contain the source code for all of the complete programs that are found within the textbook. They are available on the Instructor's ToolKit CD-ROM and also as a free download for instructors and students from the publisher's web site: www.jbpub.com/disks. The programs from all the Programming Examples, plus several programs that appear in the chapter bodies, are included. Fragments or snippets of program code are not included nor are the solutions to the chapter-ending "Programming Problems." These program files can be viewed or edited using any standard text editor, but in order to compile and run the programs, a C++ compiler must be used.

Integrated Web Site This Web site features integrated Web Links from the textbook, the complete programs from the text, and Appendix D entitled "Using this Book with a Prestandard Version of C++," which describes the changes needed to allow the programs in the textbook to run successfully with a prestandard compiler.

Student Lecture Companion: A Note-Taking Guide Designed from the PowerPoint presentations developed for this text, the Student Lecture Companion is an invaluable tool for learning. The notebook is designed to encourage students to focus their energies on listening to the lecture as they fill in additional details. The skeletal outline concept helps students organize their notes and readily recognize the important concepts in each chapter.

A Laboratory Course in C++, Second Edition Written by Nell Dale, this lab manual follows the organization of the second edition of the text. The lab manual is designed to allow the instructor maximum flexibility and may be used in both open and closed laboratory settings. Each chapter contains three types of activities: Prelab, Inlab, and Postlab. Each lesson is broken into exercises that thoroughly demonstrate the concept covered in the chapter. A disk that contains the programs, program shells (partial programs), and data files accompanies the lab manual.

Acknowledgments

We would like to thank the many individuals who have helped us in the preparation of this second edition. We are indebted to the members of the faculties of the Computer Science Departments at the University of Texas at Austin, the University of Massachusetts at Amherst, and the University of Wisconsin–La Crosse.

We extend special thanks to Jeff Brumfield for developing the syntax template metalanguage and allowing us to use it in the text.

For their many helpful suggestions, we thank the lecturers, teaching assistants, consultants, and student proctors who run the courses for which this book was written, and the students themselves.

We are grateful to the following people who took the time to review the manuscript for the parent textbook, *Programming and Problem Solving with C++, Second Edition:* J. Ken Collier, Northern Arizona State; Lee Cornell, Mankato State University; Charles Dierbach, Towsen University; Judy Etchison, Collin County Community College; David Galles, University of San Francisco; Susan Gauch, University of Kansas; Wagar Haque, University of Northern British Columbia; Ilga Higbee, Black Hawk College; Jeanine Ingber, University of New Mexico; Paula Jech, Pennsylvania State University; Hikyoo Koh, Lamar University; I. Stephen Leach, Florida State University; Joseph Marti, College of the Canyons; Kenrick Mock, Oregon State University; Viera Proulx, Northeastern University; Howard Pyron, University of Missouri–Rolla; Dennis Ray, Old Dominion University; Sujan Sarkar, Santa Rosa Junior College; Lynn Stauffer, Sonoma State University; Greg Steuben, Rensselaer Polytechnic Institute.

We also thank Bobbie Lewis and Mike and Sigrid Wile along with the many people at Jones and Bartlett who contributed so much, especially J. Michael Stranz, Amy Rose, Jennifer Jacobson, Anne Spencer, and W. Scott Smith.

Anyone who has ever written a book—or is related to someone who has—can appreciate the amount of time involved in such a project. To our families—all the Dale clan and the extended Dale family (too numerous to name); to Lisa, Charlie, and Abby; to Anne, Brady, and Kari—thanks for your tremendous support and indulgence.

N. D.
C. W.
M. H.

contents

1 Overview of Programming and Problem Solving 1

1.1 **Overview of Programming** 2

What Is Programming? 2

How Do We Write a Program? 2

1.2 **What Is a Programming Language?** 7

1.3 **What Is a Computer?** 11

1.4 **Ethics and Responsibilities in the Computing Profession** 15

Software Piracy 15

Privacy of Data 16

Use of Computer Resources 16

Software Engineering 17

1.5 **Problem-Solving Techniques** 17

Ask Questions 18

Look for Things That Are Familiar 19

Solve by Analogy 19

Means-Ends Analysis 19

Divide and Conquer 20

The Building-Block Approach 20

Merging Solutions 21

Mental Blocks: The Fear of Starting 22

Algorithmic Problem Solving 22

Summary 22

2 C++ Syntax and Semantics, and the Program Development Process 25

2.1 The Elements of C++ Programs 26

C++ Program Structure 26

Syntax and Semantics 28

Syntax Templates 30

Naming Program Elements: Identifiers 31

Data and Data Types 33

Naming Elements: Declarations 34

Taking Action: Executable Statements 38

Beyond Minimalism: Adding Comments to a Program 43

2.2 Program Construction 44

Blocks (Compound Statements) 46

The C++ Preprocessor 47

An Introduction to Namespaces 49

2.3 More About Output 50

Creating Blank Lines 50

Inserting Blanks Within a Line 51

Programming Example: Contest Letter 53

Testing and Debugging 57

Summary 59

Quick Check 59

Exam Preparation Exercises 62

Programming Warm-up Exercises 65

Programming Problems 67

3 Numeric Types, Expressions, and Output 69

3.1 Overview of C++ Data Types 70

3.2 Numeric Data Types 70

Integral Types 71

Floating-Point Types 72

3.3 Declarations for Numeric Types 73

Named Constant Declarations 73

Variable Declarations 74

3.4 **Simple Arithmetic Expressions** 74
Arithmetic Operators 75
Increment and Decrement Operators 78
3.5 **Compound Arithmetic Expressions** 78
Precedence Rules 79
Type Coercion and Type Casting 80
3.6 **Function Calls and Library Functions** 83
Value-Returning Functions 83
Library Functions 85
Void Functions 86
3.7 **Formatting the Output** 87
Integers and Strings 87
Floating-Point Numbers 90
3.8 **Additional `string` Operations** 94
The `length` and `size` Functions 94
The `find` Function 95
The `substr` Function 97
Programming Example: Map Measurements 99
Testing and Debugging 102
Summary 103
Quick Check 103
Exam Preparation Exercises 106
Programming Warm-up Exercises 109
Programming Problems 113

4 Program Input and the Software Design Process 115
4.1 **Getting Data into Programs** 116
Input Streams and the Extraction Operator (>>) 116
The Reading Marker and the Newline Character 119
Reading Character Data with the `get` Function 120
Skipping Characters with the `ignore` Function 122
Reading String Data 123
4.2 **Interactive Input/Output** 125
4.3 **Noninteractive Input/Output** 126

4.4 **File Input and Output** 127
- Files 127
- Using Files 127
- An Example Program Using Files 130
- Run-Time Input of File Names 133

4.5 **Input Failure** 134

4.6 **Software Design Methodologies** 135

4.7 **What Are Objects?** 136

4.8 **Object-Oriented Design** 138

4.9 **Functional Decomposition** 138
- Modules 140
- A Perspective on Design 141

Programming Example: Stretching a Canvas 142

Testing and Debugging 147
- Testing and Debugging Hints 148

Summary 149
- Quick Check 150
- Exam Preparation Exercises 151
- Programming Warm-up Exercises 153
- Programming Problems 155

5 Conditions, Logical Expressions, and Selection Control Structures 157

5.1 **Flow of Control** 158
- Selection 159

5.2 **Conditions and Logical Expressions** 159
- The `bool` Data Type 159
- Logical Expressions 160
- Precedence of Operators 166
- Relational Operators with Floating-Point Types 167

5.3 **The If Statement** 168
- The If-Then-Else Form 169
- Blocks (Compound Statements) 170
- The If-Then Form 172
- A Common Mistake 173

5.4 **Nested If Statements** 174
The Dangling `else` 176
5.5 **Testing the State of an I/O Stream** 178
Programming Example: Warning Notices 180
Testing and Debugging 183
Testing in the Problem-Solving Phase: The Algorithm Walk-Through 184
Testing in the Implementation Phase 186
The Test Plan 191
Tests Performed Automatically During Compilation and Execution 192
Testing and Debugging Hints 193
Summary 195
Quick Check 195
Exam Preparation Exercises 196
Programming Warm-up Exercises 199
Programming Problems 202

6 Looping 205
6.1 **The While Statement** 206
6.2 **Phases of Loop Execution** 208
6.3 **Loops Using the While Statement** 208
Count-Controlled Loops 209
Event-Controlled Loops 209
Looping Subtasks 212
6.4 **How to Design Loops** 214
Designing the Flow of Control 215
Designing the Process Within the Loop 216
The Loop Exit 217
6.5 **Nested Logic** 217
Designing Nested Loops 219
Programming Example: Average Income by Gender 220
Testing and Debugging 223
Loop-Testing Strategy 223
Test Plans Involving Loops 224

Testing and Debugging Hints 225
Summary 227
Quick Check 227
Exam Preparation Exercises 228
Programming Warm-up Exercises 231
Programming Problems 232

7 Functions 235
7.1 Functional Decomposition with Void Functions 236
Writing Modules as Void Functions 236
7.2 An Overview of User-Defined Functions 239
Flow of Control in Function Calls 239
Function Parameters 240
7.3 Syntax and Semantics of Void Functions 241
Function Call (Invocation) 241
Function Declarations and Definitions 242
Local Variables 244
The Return Statement 245
Header Files 246
7.4 Parameters 247
Value Parameters 248
Reference Parameters 249
7.5 Designing Functions 250
Writing Assertions as Program Comments 252
Documenting the Direction of Data Flow 254
Programming Example: Comparison of Furniture-Store Sales 257
Testing and Debugging 263
The `assert` Library Function 265
Testing and Debugging Hints 266
Summary 267
Quick Check 268
Exam Preparation Exercises 269
Programming Warm-up Exercises 275
Programming Problems 277