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

教学指导与习题集

（第二版　影印版）

PROGRAMMING IN C++

Instructor's Manual Adaptation

(Second Edition)

■ Nell Dale
Chip Weems
Mark Headington

高等教育出版社
Higher Education Press

书

—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

教学指导与习题集

（第二版　影印版）

PROGRAMMING IN C++

Instructor's Manual Adaptation

(Second Edition)

Nell Dale
Chip Weems
Mark Headington

高等教育出版社

图字：01-2003-0687 号

Programming in C++: Instructor's Manual Adaptation, Second Edition
Nell Dale, Chip Weems, Mark Headington

ORIGINAL ENGLISH LANGUAGE EDITION PUBLISHED BY
Jones and Bartlett Publishers, Inc.
40 Tall Pine Drive
Sudbury, MA 01776

图书在版编目(CIP)数据

C++程序设计教学指导与习题集/(美)戴尔(Dale, N.),(美)威姆斯(Weems, C.),(美)黑丁顿(Headington, M.)编. —2版(影印版). —北京：高等教育出版社，2003.3
ISBN 7-04-012658-3

Ⅰ.C... Ⅱ.①戴...②威...③黑... Ⅲ.C语言-程序设计-高等学校-教学参考资料-英文
Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2003)第 014601 号

出版发行 高等教育出版社
社　　址 北京市东城区沙滩后街 55 号
邮政编码 100009
传　　真 010-64014048

经　　销 新华书店北京发行所
印　　刷 北京铭成印刷有限公司

开　　本 787×1092　1/16
印　　张 27.5
字　　数 510 000

购书热线 010-64054588
免费咨询 800-8[illegible]0598
网　　址 http://www.hep.edu.cn
http://www.hep.com.cn

版　　次 2003 年 3 月第 1 版
印　　次 2003 年 3 月第 1 次印刷
定　　价 35.00 元(含光盘)

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经济、科技、军事、教育和文化等产生了深刻影响。信息科学技术的迅速普及和应用，带动了世界范围信息产业的蓬勃发展，为许多国家带来了丰厚的回报。

进入 21 世纪，尤其随着我国加入 WTO，信息产业的国际竞争将更加激烈。我国信息产业虽然在 20 世纪末取得了迅猛发展，但与发达国家相比，甚至与印度、爱尔兰等国家相比，还有很大差距。国家信息化的发展速度和信息产业的国际竞争能力，最终都将取决于信息科学技术人才的质量和数量。引进国外信息科学和技术优秀教材，在有条件的学校推动开展英语授课或双语教学，是教育部为加快培养大批高质量的信息技术人才采取的一项重要举措。

为此，教育部要求由高等教育出版社首先开展信息科学和技术教材的引进试点工作。同时提出了两点要求，一是要高水平，二是要低价格。在高等教育出版社和信息科学技术引进教材专家组的努力下，经过比较短的时间，第一批引进的 20 多种教材已经陆续出版。这套教材出版后受到了广泛的好评，其中有不少是世界信息科学技术领域著名专家、教授的经典之作和反映信息科学技术最新进展的优秀作品，代表了目前世界信息科学技术教育的一流水平，而且价格也是最优惠的，与国内同类自编教材相当。

这项教材引进工作是在教育部高等教育司和高教社的共同组织下，由国内信息科学技术领域的专家、教授广泛参与，在对大量国外教材进行多次遴选的基础上，参考了国内和国外著名大学相关专业的课程设置进行系统引进的。其中，John Wiley 公司出版的贝尔实验室信息科学研究中心副总裁 Silberschatz 教授的经典著作《操作系统概念》，是我们经过反复谈判，做了很多努力才得以引进的。William Stallings 先生曾编写了在美国深受欢迎的信息科学技术系列教材，其中有多种教材获得过美国教材和学术著作者协会颁发的计算机科学与工程教材奖，这批引进教材中就有他的两本著作。留美中国学者 Jiawei Han 先生的《数据挖掘》是该领域中具有里程碑意义的著作。由达特茅斯学院 Thomas Cormen 和麻省理工学院、哥伦比亚大学的几

位学者共同编著的经典著作《算法导论》，在经历了 11 年的锤炼之后于 2001 年出版了第二版。目前任教于美国 Massachusetts 大学的 James Kurose 教授，曾在美国三所高校先后 10 次获得杰出教师或杰出教学奖，由他主编的《计算机网络》出版后，以其体系新颖、内容先进而倍受欢迎。在努力降低引进教材售价方面，高等教育出版社做了大量和细致的工作。这套引进的教材体现了权威性、系统性、先进性和经济性等特点。

教育部也希望国内和国外的出版商积极参与此项工作，共同促进中国信息技术教育和信息产业的发展。我们在与外商的谈判工作中，不仅要坚定不移地引进国外最优秀的教材，而且还要千方百计地将版权转让费降下来，要让引进教材的价格与国内自编教材相当，让广大教师和学生负担得起。中国的教育市场巨大，外国出版公司和国内出版社要通过扩大发行数量取得效益。

在引进教材的同时，我们还应做好消化吸收，注意学习国外先进的教学思想和教学方法，提高自编教材的水平，使我们的教学和教材在内容体系上，在理论与实践的结合上，在培养学生的动手能力上能有较大的突破和创新。

目前，教育部正在全国 35 所高校推动示范性软件学院的建设和实施，这也是加快培养信息科学技术人才的重要举措之一。示范性软件学院要立足于培养具有国际竞争力的实用性软件人才，与国外知名高校或著名企业合作办学，以国内外著名 IT 企业为实践教学基地，聘请国内外知名教授和软件专家授课，还要率先使用引进教材开展教学。

我们希望通过这些举措，能在较短的时间，为我国培养一大批高质量的信息技术人才，提高我国软件人才的国际竞争力，促进我国信息产业的快速发展，加快推动国家信息化进程，进而带动整个国民经济的跨越式发展。

教育部高等教育司

二〇〇二年三月

PREFACE

This *Instructor's Guide* has been developed to assist both instructors who are teaching the CS1 or C101 course with C++ for the first time and those who have taught the course previously. We hope that even the most experienced of instructors will find something new and useful in the material presented here.

ORGANIZATION OF THIS GUIDE

Each chapter of the *Instructor's Guide* corresponds to one chapter in the text. Every chapter is organized into seven sections:

1. Chapter Goals
2. Chapter Outline
3. General Discussion
4. The Hard Parts
5. Exercise Answers
6. Exam Questions
7. Answers to Exam Questions

Chapter Goals

The learning objectives listed at the beginning of each chapter of the text are repeated here. (In the text, for each goal there is a corresponding Quick Check question in the end-of-chapter exercises.)

Chapter Outline

An outline of each chapter's contents is provided to aid in planning lectures.

General Discussion

This section focuses on the most important new concepts that students must understand in order to complete the chapter successfully.

The Hard Parts

In each chapter, certain topics or details routinely cause problems for students. Based on classroom experience, this section identifies the problem areas and offers techniques, tips, analogies, and anecdotes that address the difficult material. The Hard Parts section gives us a chance to share these suggestions with you, for you to use or ignore as you see fit.

Exercise Answers

The text contains answers to roughly half of the Exam Preparation and Programming Warm-Up exercises. This section contains the rest of the answers.

Exam Questions

This *Instructor's Guide* provides over 800 test questions, averaging about 55 per chapter. The questions are objective in nature and are organized into three formats: True/False, Multiple Choice, and Fill-In. An electronic version of the questions and answers, along with other Instructor's Guide content, is available on an Instructor's Toolkit CD-ROM upon request from your Jones and Bartlett representative.

Answers to Exam Questions

The final section of each chapter is an answer key for the exam questions.

COURSE ORGANIZATION

Programming in C++, Second Edition is appropriate as a text for a self-study course. More typical, however, is a lecture-oriented course structure. All but four of the chapters (6, 11, 12, and 14) may be covered in one week of either three 60-minute lectures or four 45-minute lectures. (This guide assumes the former but is easily adapted to the latter.)

In a lecture course where the group is large, it is desirable to have a lab or discussion section associated with the course. The class should be divided into lab groups consisting of roughly 15 to 25 students. Each lab section typically is administered by a teaching assistant (T.A.) who leads the discussion. Depending on the number of lab sections, more than one T.A. may be required. If possible, it's best to have each T.A. be responsible for no more than three lab sections. With a very large class, the T.A. is usually the students' primary contact for technical assistance and course administration. Although lab sections may have a purely lecture- or discussion-oriented format, there are great benefits to be reaped if machine access is available in the lab room. This permits the students to run through some hands-on exercises, with the T.A. present to answer questions and offer advice. Either way, it is important that students be given the opportunity to ask questions and discuss problems in a small-group environment. In a lecture course where the class is small, it may be desirable to schedule an extra period to allow time for this type of interaction to take place.

Below is a suggested time frame for each chapter of the text:

Chapter	*Number of weeks*	*Chapter*	*Number of weeks*	*Chapter*	*Number of weeks*
1	1	**6**	2	**11**	1.5
2	1	**7**	1	**12**	1.5
3	1	**8**	1	**13**	1
4	1	**9**	0.5	**14**	1.5
5	1	**10**	1	**15**	1

Coverage of material in the text will depend on the type of system employed at a particular school. At a semester school, for example, a term is roughly 16 weeks in length. This allows coverage of Chapters 1 through 13 (Array-Based Lists). Alternatively, you could quicken the pace through the following chapters: 6, 10 (skimming floating point representation), 11 (skimming structs), and 12 (skimming multidimensional arrays). Doing so would allow coverage of Chapter 14 (Object-Oriented Software Development).

In a school that operates on a quarter system, each quarter is about 11 weeks. Instructors can cover Chapters 1 through 9 (Additional Control Structures) in the first quarter and, after picking up with a review of Chapter 9, finish the rest of the book in the second quarter.

READERS' COMMENTS

We greatly appreciate the feedback that we receive from instructors, teaching assistants, and students. We're already planning the next edition of the text and welcome your comments, whether favorable or unfavorable. Our goal, after all, is to make this the best *Programming and Problem Solving with C++* that we possibly can.

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

CONTENTS

Chapter 1

OVERVIEW OF PROGRAMMING AND PROBLEM SOLVING

CHAPTER GOALS

- To understand what a computer program is.
- To be able to list the basic stages involved in writing a computer program.
- To understand what an algorithm is.
- To learn what a high-level programming language is.
- To be able to describe what a compiler is and what it does.
- To understand the compilation and execution processes.
- To learn what the major components of a computer are and how they work together.
- To be able to distinguish between hardware and software.
- To learn about some of the basic ethical issues confronting computing professionals.
- To be able to choose an appropriate problem-solving method for developing an algorithmic solution to a problem.

CHAPTER OUTLINE

I. Overview of Programming
   A. What is Programming?
   B. How Do We Write a Program?
II. What is a Programming Language?
III. What Is a Computer?
IV. Ethics and Responsibilities in the Computing Profession
   A. Software Piracy
   B. Privacy of Data
   C. Use of Computer Resources
   D. Software Engineering
V. Problem-Solving Techniques
   A. Ask Questions
   B. Look for Things That Are Familiar
   C. Solve by Analogy
   D. Means-Ends Analysis
   E. Divide and Conquer
   F. The Building-Block Approach
   G. Merging Solutions
   H. Mental Blocks: The Fear of Starting

I. Algorithmic Problem Solving
VI. Summary

GENERAL DISCUSSION

In the first four chapters we introduce the basic concepts and terminology of computer science and computer programming. This may seem like a slow-paced introduction, but we have found that if we take more time at the beginning to ensure that students comprehend the basics, it's possible to move much more quickly in later chapters. The net result is that more material is covered overall.

This chapter introduces students to the terminology of computers, the concepts and methodologies of problem solving and algorithms, the essentials of how computers work, and some ethical issues in computer science.

The most important concept in this chapter is the algorithm: a step-by-step procedure for solving a problem in a finite amount of time. Stress that algorithms encompass far more than computer programs. The students should understand that they are learning to write algorithms and that the computer is simply a fast and flexible tool for implementing algorithms.

The material in this chapter is usually covered in one week. The first lecture typically covers course administration, basic terminology, how to get access to machines, and so on. In the second lecture, you can discuss algorithms, implementations, compilation, execution, and computer organization. The third lecture introduces the different problem-solving techniques and ethical issues in computer science.

THE HARD PARTS

Source Programs, Object Programs, and the Compiler

It is important that students understand the concept of high-level language versus machine language and the role of the compiler in bridging the gap. Students who know only BASIC may have used an interpreter rather than a compiler. The process of translating the entire source program into object code instead of translating and executing one instruction at a time will be new to them. You should tie the fundamentals of the operating system and computer architecture into this discussion so that the students can see how the whole system works together to compile and execute a program.

Because the students won't be writing and running their first C++ program until Chapter 2, the discussion of compilation and execution is bound to be somewhat abstract to them. You may want to tell the students that the compilation process will become clearer to them in the next chapter and that they should refer back to Chapter 1 at that time.

Algorithms and Problem Solving

Students must be able to look at a simple problem and see where to begin to solve it. It will take them years to fully develop this skill, but some initial examples and practice will give them the confidence to keep going. The key concept here is to realize that solutions to programming problems involve writing out the steps in computing the solution, rather than computing the solution itself. Many students will not fully appreciate the difference until the idea of input is introduced in Chapter 4.

From the start, it should be made clear that problem solving and algorithm design are techniques that apply to many areas other than programming. A good way to demonstrate this is to describe some algorithms that are encountered in everyday life. These might include a musical score, instructions for assembling a stereo system, instructions for knitting a sweater, or a recipe for a chocolate cake. Each algorithm may be characterized as having an author, a processor, and something that is being processed or produced. In the case of a musical score, the author is a composer, the processor is one or more musicians, and the thing being processed or produced is sound. You then can introduce programs as just another type of algorithm—their authors are programmers, the processor is a computer, and the thing being processed or produced is data. You should point out to the students that they will be learning how to be authors of programs in much the same way that musicians learn how to be composers of music.

Each of the sample algorithms also has its own language. The stereo assembly instructions use jargon, the musical score uses standard musical notation, the knitting instructions use a special notation (K1P2 means knit one, purl two), and the cake recipe uses various abbreviations and special terms. These examples lead naturally into a discussion of a programming language as just another language for expressing an algorithm. Students will feel much more comfortable with the idea of learning a programming language if they can equate the experience with learning to read music or learning knitting instructions.

If possible, you should lead students through a discussion of a nonprogramming problem for which an algorithmic solution can be developed. For example, they could develop an unambiguous set of instructions (an algorithm) for getting to the nearest fast-food restaurant. If this exercise is done in a small-group setting, you can call on students to supply the steps necessary to perform the task. You should make an effort to point out any ambiguous instructions, so that the students begin to get an appreciation for the precision with which they will be required to specify the steps in their programs.

EXAM QUESTIONS

True/False

1. There is only one unique general solution (algorithm) for a given problem.

2. All computer programs are algorithms.

3. All algorithms can be implemented as computer programs.

4. In a computer, data is represented electronically by pulses of electricity.

5. RAM stands for random access memory.

6. The term *software* refers to the physical components of a computer.

7. The compiler is a program that translates a high-level language program into machine code.

8. The two components of the central processing unit (CPU) are the arithmetic/logic unit and the control unit.

9. Magnetic tape drives and floppy disk drives are examples of auxiliary (secondary) storage devices.

10. In the "solve by analogy" technique, you solve a problem by modifying the solution to a similar problem.

Multiple Choice

11. Which one of the following is *not* one of the three major phases in the life cycle of a computer program?
    a. the problem-solving phase
    b. the management phase
    c. the implementation phase
    d. the maintenance phase

12. Which of the following is the *first* step in the *problem-solving phase* of a computer program's life cycle?
    a. Translate the general solution into code.
    b. Write a general solution for the problem.
    c. Test the general solution.
    d. Analyze the problem.
    e. Test the solution on a computer.

13. Which of the following is the *second* step in the *problem-solving phase* of a computer program's life cycle?
    a. Translate the general solution into code.
    b. Write a general solution for the problem.
    c. Test the general solution.
    d. Analyze the problem.
    e. Test the solution on a computer.

14. Which of the following is the *first* step in the *implementation phase* of a computer program's life cycle?
    a. Translate the general solution into code.
    b. Write a general solution for the problem.
    c. Test the general solution.
    d. Analyze the problem.
    e. Test the solution on a computer.

15. Which of the following is the *second* step in the *implementation phase* of a computer program's life cycle?
    a. Translate the general solution into code.
    b. Write a general solution for the problem.
    c. Test the general solution.
    d. Analyze the problem.

e. Test the solution on a computer.

16. The following series of steps is not an algorithm. How would you correct it?

Putting on a Pair of Athletic Shoes

Step 1. Put on one shoe.
Step 2. Tie the laces.
Step 3. Repeat.

a. Exchange steps 1 and 2.
b. Exchange steps 2 and 3.
c. Change step 3 to "Repeat once."
d. Change step 1 to "Put on both shoes."

17. Inside a computer, a single character such as the letter A usually is represented by a:
a. bit
b. byte
c. nibble
d. word

18. Which of the following translates a program written in a high-level language into machine code?
a. a mouse
b. a compiler
c. an operating system
d. an editor

19. Which of the following most closely resembles human language?
a. a high-level language
b. a machine language
c. a RAM

20. Which of the following terms describes the repetition of statements (instructions) while certain conditions are met?
a. sequence
b. selection
c. looping
d. subprogram

21. Which of the following terms describes the execution of different statements (instructions) depending on certain conditions?
a. sequence
b. selection
c. looping
d. subprogram

22. Which of the following terms describes the execution of a series of statements (instructions) one after another?
    a. sequence
    b. selection
    c. looping
    d. subprogram

23. Of the following components of a computer, which one stores data and instructions?
    a. input device
    b. output device
    c. arithmetic/logic unit
    d. control unit
    e. memory unit

24. Of the following components of a computer, which one performs computations?
    a. input device
    b. output device
    c. arithmetic/logic unit
    d. control unit
    e. memory unit

25. Of the following components of a computer, which one assures that program instructions are executed in the proper sequence?
    a. input device
    b. output device
    c. arithmetic/logic unit
    d. control unit
    e. memory unit

26. Of the following components of a computer, which one fetches the next instruction from RAM during program execution?
    a. input device
    b. auxiliary storage device
    c. arithmetic/logic unit
    d. control unit
    e. memory unit

27. Of the following components of a computer, which one presents the results of the processing to the outside world?
    a. input device
    b. output device
    c. arithmetic/logic unit
    d. control unit
    e. memory unit

28. Which problem-solving technique involves the breaking up of a large problem into smaller units that are easier to handle?
    a. divide and conquer

b. means-ends analysis
c. solving by analogy
d. the building-block approach
e. merging solutions

29. Which problem-solving technique involves integrating existing solutions on a step-by-step basis into a complete solution?
a. divide and conquer
b. means-ends analysis
c. solving by analogy
d. the building-block approach
e. merging solutions

30. Which problem-solving technique involves defining the beginning and ending states of a problem, then comparing different methods for getting between the states?
a. divide and conquer
b. means-ends analysis
c. solving by analogy
d. the building-block approach
e. merging solutions

31. Which problem-solving technique involves recognizing that the problem you are working on is similar to one that you have worked on before?
a. divide and conquer
b. means-ends analysis
c. solving by analogy
d. the building-block approach
e. merging solutions

Fill-In

32. A general solution, or algorithm, is written during the _____ phase of a computer program's life cycle.

33. Coding of an algorithm takes place during the _____ phase of a computer program's life cycle.

34. Modifications are made to an existing computer program during the _____ phase of the program's life cycle.

35. A(n) _____ is a step-by-step procedure for solving a problem in a finite amount of time.

36. A(n) _____ is a set of rules, symbols, and special words used to construct a computer program.

37. _____ is the written text and comments that make a program easier for others to understand, use, and modify.

38. _____ is information that has been put into a form that a computer can use.

39. A single binary digit (that is, a single 1 or 0) is called a(n) _____.

40. A sequence of 8 bits is known as a(n) _____.

41. _____ is the language made up of binary-coded instructions that are used directly by the computer.

42. A(n) _____ is a program that translates a high-level language program into machine code.

43. A program written in a high-level programming language is called the _____ program.

44. A(n) _____ is an input, output, or auxiliary storage device attached to a computer.

45. The _____ program is the machine language version of a source program.

46. The _____ is the set of programs that manages all of a computer's resources.

47. A(n) _____ is a shared boundary that allows independent systems to meet and act on or communicate with each other.

48. In the "_____" problem-solving technique, you recognize any subtasks that have been solved before and use those as solutions to part of the problem.

49. In the "_____" problem-solving technique, you define the beginning and ending states of the problem, then compare different methods for getting between them.

50. In the "_____" problem-solving technique, you break the problem up into smaller pieces that have been solved before, then tie the pieces together into a complete solution.