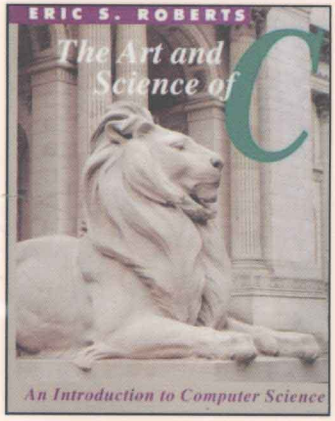
HZ BOOKS
华章科技

PEARSON

C语言经典译丛

# C语言的科学和艺术




## The Art and Science of C

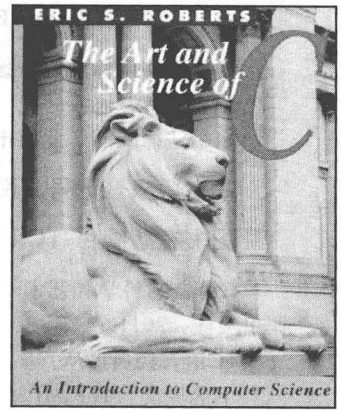
A Library-Based Introduction to Computer Science

（美）Eric S. Roberts 著

翁惠玉 张冬茉 杨鑫 蒋文新 译

机械工业出版社
China Machine Press

# C语言的科学和艺术




## The Art and Science of C

A Library-Based Introduction to Computer Science

（美）Eric S. Roberts 著

翁惠玉 张冬茉 杨鑫 蒋文新 译

机械工业出版社
China Machine Press

本书是计算机科学的经典教材，介绍了计算机科学的基础知识和程序设计的专门知识。本书以介绍ANSI C为主线，不仅涵盖C语言的基本知识，而且介绍了软件工程技术以及如何应用良好的程序设计风格进行开发等内容。本书采用了库函数的方法，强调抽象的原则，详细阐述了库和模块化开发。此外，本书还利用大量实例讲述解决问题的全过程，对开发过程中常见的错误也给出了解决和避免的方法。本书既可作为高等院校计算机科学入门课程及C语言入门课程的教材，也是C语言开发人员的极佳参考书。



**本书版权登记号：图字：01-2004-1204**

**图书在版编目（CIP）数据**

C语言的科学和艺术 /（美）罗伯茨（Roberts, E. S.）著；翁惠玉等译. —北京：机械工业出版社，2011.6

（C语言经典译丛）

书名原文：The Art and Science of C: A Library-Based Introduction to Computer Science

ISBN 978-7-111-34775-0

Ⅰ. C…　Ⅱ. ① 罗…　② 翁…　Ⅲ. C语言－程序设计　Ⅳ. TP312

中国版本图书馆CIP数据核字（2011）第091240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：秦　健

北京京北印刷有限公司印刷

2011年6月第1版第1次印刷

186mm × 240mm · 35印张

标准书号：ISBN 978-7-111-34775-0

定价：79.00元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

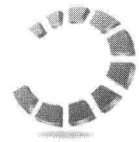
客服热线：（010）88378991；88361066

购书热线：（010）68326294；88379649；68995259

投稿热线：（010）88379604

读者信箱：hzjsj@hzbook.com

# 译者序



随着计算机产业的迅速发展，对计算机专业人才的需求也日益迫切。而程序设计是所有计算机专业人才必备的基础知识和技能。俗话说“万事开头难”，如何使学生顺利地进入程序设计的大门，如何熟悉和精通程序设计，也是计算机专业教学的难题。

本书是一本计算机科学的经典教材，是作者二十多年来从事计算机教学的经验的总结，它提供了丰富的计算机科学的基础知识和程序设计的专门知识。本书具有鲜明的特色。首先，用ANSI C作为教学语言。C语言是目前使用最广泛的教学语言，选用C语言可以使学生毕业后很快就能投入实际工作，并为学习C++和面向对象的语言铺平了道路。第二，采用了基于库函数的方法，强调抽象的原则。本书相当详细地介绍了库和模块化开发，介绍了如何通过库隐藏程序的复杂性，这些是现代程序设计的基本概念。第三，在程序设计中最重要的是从陈述问题过渡到解决问题，本书以通俗易懂的方式讲述了这一过程，使学生能轻松而有趣地学习程序设计。

程序设计既是一门科学，也是一门艺术。学习良好的程序设计需要掌握很多知识，而不只是记住一组规则。必须通过实践以及阅读其他程序来学习。本书包括大量的程序实例，这些实例说明了如何用C语句建立一个完整的程序，如何培养良好的程序设计风格。每章都用丰富的复习题作为知识点的总结，并包含大量的程序设计练习让读者自己动手做更多的程序设计项目。

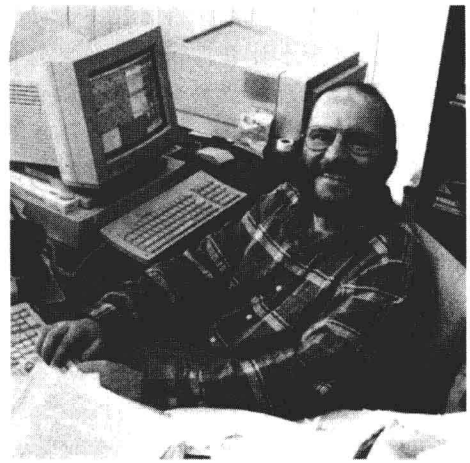
正是因为本书具有的上述优点，我们认为把本书译成中文能让更多的学生从中获益，从而打下扎实的程序设计的基础。

参加本书翻译工作的有翁惠玉、张冬茉、杨鑫和蒋文新，由翁惠玉对全书进行审校。本书也是上海交通大学《程序设计》课程所选用的教材。在翻译过程中得到了整个课程小组十多位教师的大力帮助，在此表示衷心的感谢。由于时间和水平的限制，书中难免有错漏之处，敬请读者指正。

译 者

# 前　言

## 作者简介



Eric S. Roberts第一次教授计算机科学入门课程是在20多年前，当时他还是一名哈佛学子。1980年获得博士学位以来，Roberts先后在哈佛、韦尔斯利、斯坦福大学教授计算机科学课程。在斯坦福大学，Roberts担任计算机科学系副主任，并负责本科生的计算机科学课程。虽然Roberts也教授计算机科学的高级课程并作一些研究工作，但他最大的乐趣还是向初学者展示计算机的强大功能。

在斯坦福大学任教的同时，Roberts自1990年起还担任计算机专业协会主席。这是一个由计算机专业人士组成的公益团体，在全美有22个分会，2000名会员。计算机在多方面影响着我们的社会。学习有关计算机的技术固然重要，但确保计算机服务于社会也很重要。

若读者发现本书中有表述不清楚的地方或错误之处，恳请不吝赐教。请通过Roberts的电子邮箱ericr@aw.com与他联系。

## 致学生

欢迎你！拿起这本书，你就迈进了计算机科学的世界——这门学科出现在半世纪以前，现在却成为这个时代最具生机和活力的学科之一。

在几十年的发展过程中，计算机几乎使所有领域中看似不可能的事情成为可能。由于计算机可在瞬间将信息传递到任何地方，所以今天的企业家能以空前的规模经营跨国公司。由于计算机可进行必要的、但人工很难完成的计算，科学家才能解决许多问题。电影人利用计算机制作出更具感染力的视觉效果。由于计算机能处理医学中大量的信息处理，因此医生能对患者的病情做出更精确的诊断。

计算机技术正在飞速发展。目前我们已经看到的优势与新的世纪将要经历的发展相比肯定将相形见绌。最近50年，计算机已经对世界产生了深远影响，在新的世纪亦将如此。今日的学生将会是执行这项伟大的工程的中流砥柱。要做到这一点，就必须懂得如何使用计算机。

和其他值得掌握的技能一样，理解计算机的工作原理以及学会怎样控制它们是需要花费时间的。这一切不可能一蹴而就，必须从某个起点开始循序渐进。2500年前，中国的哲学家老子曾说过："千里之行，始于足下"。本书就是一个很好的起点。

然而对很多人来说，万事开头难。许多学生在计算机面前束手无策，认为计算机科学超出了他们的理解范围。可是基本的程序设计并不需要具备高等数学和电子学的知识。在程序设计中，最重要的是能否从陈述问题过渡到解决问题。要做到这一点，就必须以逻辑方式考虑问题。训练自己用计算机能够理解的方式表达自己的逻辑。最重要的是，不要被困难和挫折压倒，要坚持到底。若能坚持下来，就会发现解决问题是件多么令人兴奋的事情，它所带来的喜悦足以让你忘却学习过程中遇到的任何挫折。

本书旨在教授程序设计基础和C语言基础。C语言是当今计算机产业中处于主导地位的程序设计语言。本书不但介绍了程序设计中的"为什么"，还介绍了"如何做"，使读者对程序设计有总体的印象。为使读者避免出现那些阻碍学习的错误，本书在结构上做出了精心安排，可以帮助读者掌握重点。接下来将总结本书在结构上的一些独具匠心之处，并说明如何在学习过程中高效地利用本书。

## 本书的特色

为了使本书在学习C语言的过程中发挥最大的作用，首先要充分了解本书的一些特色。

1) 本书的每一章都包含一些指导读者学习以及掌握主题的材料。

- 学习目标中列出了该章包含的关键内容。因为每个目标都与一个具体的技能相关，所以该章的学习目标有助于你评估自己对基本知识的掌握程度。

### 第1章 概 述

[The Analytical Engine offers] a new, a vast, and a powerful language . . . for the purposes of mankind.

— Augusta Ada Byron, Lady Lovelace, *The Sketch of the Analytical Engine Invented by Charles Babbage*, 1843

**学习目标**

- 理解硬件和软件的区别。
- 认识问题求解是计算机科学的一个重要组成部分。
- 理解术语算法的含义。
- 理解编译器在高级程序设计语言和低级机器语言之间所起的翻译器的作用。
- 认识程序设计错误的主要类型。
- 认识软件维护的重要性和使用良好的软件工程实践方法的重要性。

在计算机技术高速发展的今天，很难想像在1940年计算机还不存在。如今，计算机随处可见，大家都会说，我们生活在计算机时代。

### 1.1 计算简史

从某种意义上说，计算的使用古已有之。许多早期数学家致力于解决重要的实际问题的计算，如观察兽群中动物的数量，计算小块土地的面积或记录商业交易等。这些活动要求人们开发新的计算技术，有时甚至要求人们发明新机器来帮助完成计算过程。例如，在亚洲使用了几千年的算盘，它是由在杆上滑动的小珠组成，这种计数装置大约在公元前2000年就出现了。

- 小结部分详细地描述了你应该学到的与学习目标部分相关的知识。

## 小结

第3章从宏观角度展示了程序设计的过程，强调了问题的求解。在此过程中，以非正式的方法介绍了一些控制语句。本章详细地研究了这些控制语句的工作细节。此外，还介绍了一个新的数据类型——*布尔型数据*。尽管这种数据类型只有TRUE和FALSE两个值，但是能高效地使用它对成功的程序设计十分重要，因此应该多加练习。

本章中还介绍了几个新的运算符，在这里回顾一下已学过的运算符间的优先级关系是有好处的。表4-1总结了这些信息，所有的运算符是按优先级递减的顺序列出的。

**表4-1 本章中出现过的运算符的优先级表**

| 运 算 符 | 结 合 性 |
|---|---|
| 一元 - ++ -- ! (类型转换) | 从右到左 |
| * / % | 从左到右 |
| + - | 从左到右 |
| < <= > >= | 从左到右 |
| == != | 从左到右 |
| && | 从左到右 |
| \|\| | 从左到右 |
| ?: | 从右到左 |
| = *op*= | 从右到左 |

本章的重要知识点包括：

- *简单语句*由表达式跟上分号构成。
- “=”是C语言中用来赋值的运算符。因此，赋值是一个合法的表达式。它可以以*嵌套赋值*和*多重赋值*的形式出现。
- 多个独立的语句可以组合为*复合语句*，通常称为*程序块*。
- 控制语句分成两类：*条件*和*迭代*。
- genlib库定义了一种bool数据类型，用它来表示布尔型数据。bool型数据只有TRUE和FALSE两个值。
- 使用*关系运算符*（<, <=, >, >=, ==和!=）能够产生布尔值，然后可以使用*逻辑运算符*（&&, ||和!）连接关系表达式。
- 逻辑运算符&&和||是从左到右计算的，一旦能确定整个表达式的值就停止计算。这种行为叫做*简化求值*。
- 当有些代码仅在特定情况下才执行或程序需要在两条执行路径之间选择其一时可以使用if语句。

- 有些问题有这样的结构：在情况1下完成操作1，情况2下完成操作2，依此类推。此时就可以使用**switch**语句。
- **while**语句用来指定在一定条件满足的情况下重复执行某些操作。
- **for**语句指示一定次数的重复操作，它在每个周期中要更新下标变量的值。

2) 程序设计既是一门科学，也是一门艺术。形成良好的程序设计风格需要掌握很多知识，而不只是记住一组规则。你必须通过实践并阅读其他程序来不断学习。书中在介绍程序设计时的一些特色可以为你提供帮助。

- 本书包括大量的程序实例，这些实例说明了如何用C语句建立一个完整的程序。这些实例也可作为你自己设计程序时的模型。在许多情况下，你可以对书中的程序作简单的修改来解决一个新的程序设计问题。

```
/*
 * File: liftoff.c
 * ---------------
 * Simulates a countdown for a rocket launch.
 */

#include <stdio.h>
#include "genlib.h"

/*
 * Constant: StartingCount
 * -----------------------
 * Change this constant to use a different starting value
 * for the countdown.
 */

#define StartingCount 10

/* Main program */

main()
{
    int t;

    for (t = StartingCount; t >= 0; t--) {
        printf("%2d\n", t);
    }
    printf("Liftoff!\n");
}
```

图4-7 liftoff.c

- 实例输出具有统一的格式，都是用一个带圆角的方框模拟计算机的屏幕。输入用黑体表示。

```
This program sums the digits in an integer.
Enter a positive integer: 1729↵
The sum of the digits is 19
```

• 语法框总结C语法的主要规则，对关键程序设计概念作简单的回顾。

**语法：多行if语句**

```
if (condition) {
    statements;
}
```

其中：*condition*是将要测试的布尔值。

*statements*是当条件为TRUE时将要执行的一个语句块。

3) 所有的程序员，即使是最好的程序员，都会犯错误。在程序中找出这些错误是非常重要的。以下这些特色有助于你培养这些技能。

• 为了帮助你识别和纠正逻辑错误，本书包括了一些说明某些典型错误的程序。为了确保你不会把这些程序作为模型，本书在这些不正确的程序上面叠印了一个苍蝇的图像。

```
/*
 * File: balance2.c
 * ----------------
 * This file contains a buggy second attempt at a program to
 * balance a checkbook.
 */

#include <stdio.h>
#include "genlib.h"
#include "simpio.h"

main()
{
    double entry, balance;

    printf("This program helps you balance your checkbook.\n");
    printf("Enter each check and deposit during the month.\n");
    printf("To indicate a check, use a minus sign.\n");
    printf("Signal the end of the month with a 0 value.\n");
    printf("Enter the initial balance: ");
    balance = GetReal();
    while (TRUE) {
        printf("Enter check (-) or deposit: ");
        entry = GetReal();
        if (entry == 0) break;
        balance += entry;
        if (balance < 0) {
            printf("This check bounces.  $10 fee deducted.\n");
            balance -= 10;
        }
        printf("Current balance = %g\n", balance);
    }
    printf("Final balance = %g\n", balance);
}
```

• 常见错误对所有初级程序员常犯的错误给出了提示，并说明如何避免这些错误。代码中的错误行用苍蝇图案加以突出，旁边给出相应的说明。

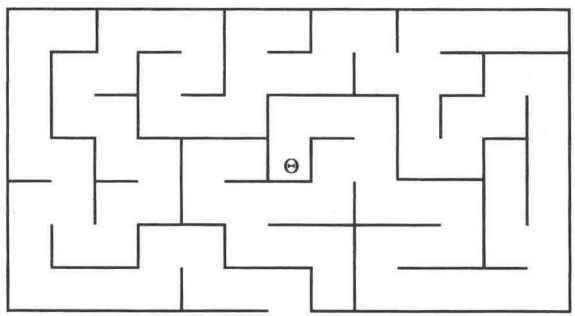
> 在大多数情况下，逻辑表达式不至于复杂到必须用真值表来计算的程度。唯一容易引起混淆的是!或!=和&&或||一起出现的情况。当说起某某情况不为真时（即需要用到!或!=的情况），日常用语和数学逻辑表达有时是相悖的，因此需要格外小心避免犯错。比如，在程序中要表达这样一个意思：*x*不等于2或3。通过这个条件测试语言表述，新程序员很可能会写出以下语句：
>
> ```
> if (x != 2 || x != 3) . . .          (这是错误的)
> ```
>
> 如果用数学观点来观察这个条件测试，会发现只要x不等于2或只要x不等于3，if测试中的表达式均为TRUE。无论x取什么值，其中一个表达式必定为真，因为如果x为2，则它不可能同时为3，反之亦然。因此上面写出的if测试将永远成功。
>
> **常见错误** 当&&和||运算符与涉及!和!=运算符的关系测试一起使用时，必须非常小心。自然语言在逻辑上有一些模糊，而程序设计要求非常精确。

4) 学习程序设计需要大量的实践。为了保证你能进行必要的实践，每章都包括了大量的练习和问题，测试你对各章内容的掌握程度。

• 每章最后都有大量的复习题，其中包括对该章内容的回顾、代码的改编以及调试的练习。

> **复习题**
>
> 1. 判断题：本章包含了你需要知道的有关接口的所有信息。
> 2. 定义下列术语：接口、软件包、抽象、实现者、客户。
> 3. 实现者和客户的观点有何区别?
> 4. 在C语言中，接口是如何表示的?

• 程序设计练习让你自己动手进行更多的程序设计项目。

> **程序设计练习**
>
> 1. 虽然本章重点介绍数学的算法，但希腊人也非常喜欢研究一些其他类型的算法。例如，在希腊神话中，雅典的特修斯拿着一个线球，一边走一边放线，然后沿着线回到了出口，从人身牛头怪物的迷宫中逃了出来。特修斯的策略表示了从迷宫中出逃的一个算法，但这并不是解决这个问题的唯一算法。例如，如果迷宫内部没有回路，那么总是可以根据*右手法*则逃出来，即右手总是贴在墙上。这种方法有时会使你从原路返回，但能保证最终找到出口。
>
> 例如，假设特修斯在迷宫中的位置用希腊字母Θ表示：
>
> 
> 

## 致教师

从1991～1992年，斯坦福大学决定调整计算机科学系的入门课程，使用C语言来代替原先使用的Pascal语言。选择ANSI C的主要原因如下：

- 学生要求学习更为实用的语言。雇主期望学生们对业界使用的工具有一定的使用经验，而当今业界使用最广泛的语言工具是C和C++。在招聘中，使用Pascal语言的程序员的就业机会越来越少，于是有的学生开始质疑教育的初衷。
- 许多基于Pascal语言的课程要求学生使用今后很可能用不到的Pascal语言编程。我们发现许多学生在放弃使用Pascal语言转而学习更现代的语言时，往往忘记曾经学过的程序设计风格和软件工程原则。现在改用学生们会继续使用的语言来教授这些技巧，结果发现他们能写出更好的程序。
- 许多高级的计算机科学课程，尤其是偏系统方面的课程，要求学生使用C语言编程。在入门时就使用C语言可以使学生们在学习高级课程时有更多的经验。
- 早日学习C语言为学习C++和面向对象的语言铺平了道路。由于学生在第一年的学习后，就有了较丰富的使用C语言的经验，学校就可以更早开设教学计划中的面向对象的系统设计这门课程。
- C语言中覆盖了一些重要的主题，如模块化开发和抽象数据类型等。这在Pascal语言环境里是很难讲授的。
- 在最近五年中，在工程科学领域的编程中，C语言正在逐步取代Fortran的重要地位。

考虑到近三年的经历，作者确信选用C语言作为计算机科学的入门课程是正确的选择，这种改变将使将来的程序更为强壮。

与此同时，也要意识到在程序设计的早期课程中教授C语言并不是那么容易。C语言是为程序员设计的，并非为作为初学者的学生设计的。其中很多的特性只有在理解了大的概念框架后才有意义，而初学者往往意识不到这一点。从许多方面讲，C语言都是一门复杂的语言，向初学者讲授这门语言时必须很好地控制讲授的进度。

## 基于库函数的方法

让教师能够控制C语言所固有的复杂性是本教材的中心目标之一。而控制复杂性又是程序员的重要职责。当面对一个很复杂的、不能立即解决的问题时，可将它分解成多个部分，然后单独思考每个部分的解决方案。此外，当某个部分达到了一定的复杂度时，可以将它抽象出来，定义为一个简单的接口。这样，用户就不必仔细理解这个抽象的细节，从而简化了程序的概念结构。

这种方法也适用于程序设计的教学。为使知识更浅显易懂，本书采用了基于库函数的方法，该方法强调抽象的原则。这种方法的特征反映在以下两方面，这也是本书区别于其他入门教材之处：

1) 在开始部分详细地介绍库和模块化开发，这是现代程序设计的基本概念。第二部分集中介绍库、接口、抽象和模块化开发的知识。这样，学生在学习数组之前就可以先学会高效使用

这些技术。

2) 本书通过库函数的使用来展示它们的功能。一方面告诉学生库函数可以隐藏复杂性，另一方面向他们说明了这个概念。本书介绍了一些新的库函数以帮助学生写出一些有用的程序，但在他们能消化吸收之前并不讲述库函数所隐藏的细节内容。它们的具体实现放在后面的章节讲述，以使学生能够更深刻地意识到抽象的重要作用。

在1992年，作者曾试图只用ANSI C的库函数来讲授入门课程，但结果并不理想。每个新的主题都要求学生对C语言的其余部分很熟悉，这样便无法开展教学。例如，字符串操作尽管在概念上很简单，但在学生能灵活使用字符串之前，他们必须理解数组、指针和内存分配等机制。最优秀的学生也是到了学期末才理解，而大部分学生则根本没有明白这些概念。自从1993年初引进基于库函数的方法以来，学生理解起来更容易了，对计算机这个学科的了解也更多了。

本书使用的库函数接口和相关实现见附录B。附录B还给出了获取源代码的方法。

## 讲述顺序

本书的章节顺序与斯坦福大学的入门课程的授课顺序大致相同，只是将第17章的内容放在第二门课程中讲授。教师可以根据学生情况和授课的具体目标调整讲述顺序。以下简要地说明章节的内容以及它们之间的依赖关系。

第1章追溯计算机的发展史，描述了程序设计的过程。该章不需要读者具备程序设计知识，仅仅是为其他章提供相关背景知识。

第2章和第3章主要面向稍具或完全不具备程序设计基础的学生。这两章意图讲述一些概念性的内容，关注问题的解决而不纠缠于C语言的细节问题。初学者在面对纷繁的语法和结构时，往往全神贯注地学习规则而忽视了基本的概念，但在该阶段，对初学者来说，掌握概念比掌握规则重要得多。如果学生已经对程序设计有所了解，则这两章可以一带而过。

第2章和第3章中的方法是非正规的，重点在于培养学生解决问题的能力。虽然这两章介绍了一些基本的语句形式和控制结构，但这些语句形式和控制结构只是完成一个特定任务的习语。第4章介绍每个语句的正式结构，并详细描述其语法和语义。该章还对布尔型数据进行了广泛的讨论。

第5章开始介绍函数和过程。讲述的例子由简单到复杂。该章专门安排了一个小节讨论参数传递机制，并附有很多图表以帮助学生领会数据从一个函数向另一个函数传递的过程。该章的末尾给出一个重要的编程示例以说明逐步精化的概念。

第6章所讲述的算法的概念是计算机科学的基础，但不要求所有的学生都掌握。若读者是工程人员或有潜力的计算机专业学生，那么这些材料对他们会大有裨益，给非专业的学生授课时则不必介绍得太过深入。

作者发现，在入门课程中加入图形部分可以提高学生的学习兴趣，因此专门编写了第7章。在此阶段，学生已经学习了函数的机制，但还没有实际编写程序的欲望。让学生在屏幕上画出复杂的图形可以激发他们的这种欲望。一些主要的编程环境都实现了图形库，因此在大多数情况下都可以使用。

第8章有两个主题，这两个主题从某种意义上讲是相互独立的。第一部分介绍接口的设计标准，这对于任何一个需要理解现代程序设计原理的人来说都是至关重要的。第二部分运用这些标准建立了一个随机数函数库。虽然学习random.h的具体接口并不如学习通用的接口设计标准那样重要，但后面的一些练习是需要使用随机数库函数的。

第9章将字符串作为一个抽象的数据类型来介绍，并在一定程度上介绍了一些基于库函数的方法。利用动态字符串库，学生可以轻而易举地编写出一些复杂的字符串操作程序，即使他们并不能理解底层的表示方法。这些内容将在第14章中介绍。字符串的引入可以使学生编写出更优秀的程序。

第一遍浏览过后，读者很容易忘记第10章的真正目的，仿佛它是第9章中对字符串讨论的继续。该章的重要价值并不在于程序Latin Pig，它虽然有趣但却不实用。举这个例子的目的是使读者意识到构造一个用来区分用户输入中各个单词的扫描器接口的必要性。不单单在第一门课程，而且在其后续课程中这个扫描器模块都将显示出其实用性，它将是学生在本课程中创造的最具实用价值的工具，同时也是说明模块化重要性的一个最佳例子。

第11章～第16章分别介绍了一些基本的复合结构——数组、指针、文件、记录等，这样的介绍顺序在实际教学实践中的效果较好。考虑到作为基础语言的C语言的一些特性，为了要强调这些结构之间的联系，在介绍过数组后应尽快介绍指针。有了这些概念做基础，第14章就可以更加严密地讲述字符串数据，从而揭示出封装在抽象定义中的内容。第16章构造了一个数据驱动的教学程序，它综合运用了这些基本的数据结构，是本书中最复杂的一个程序设计示例。

第17章包含了在第一门程序设计课程中经常出现的三个重要主题：递归、抽象数据类型和算法分析。在斯坦福，这些主题在第二门课程中用1/4学期的时间讲述。若决定在第一门程序设计课程中讲述递归，强烈建议早日开始介绍，以便学生有时间去消化吸收。教师可以在第5章后直接讲述递归函数，在第6章后讲述递归算法，也可以在第12章后将递归和算法分析放在一起讲解。

## 教师资源

教师要获取本书的相关教学资源，请到华章网站www.hzbook.com上申请。

## 致谢

本书的前身是课程讲义，在此基础上进行了大量改进而形成本书，这很大程度上要归功于使用本书的读者的建议。在这里我要衷心感谢斯坦福大学的讲师们：Jon Becker、Katie Capps、Stephen Clausing、Todd Feldman、Allison Hansen、Margaret Johnson、Jim Kent、Andrew Kosoresow、Mehran Sahami和Julie Zelenski，他们在过去三年的14个班讲授了本书。此外，我还要感谢所有的班导师、助教和教务人员：Felix Baker、John Lilly、Sandy Nguyen、Bryan Rollins以及Scott Wiltamuth，他们提供了必要的后勤支持。

本书中很多好的想法来自于一个计算机入门教学的社团，这是我于1992～1993年倡导成立的。它为解决一些困难问题和对教材进行重要改进提供了一个讨论的平台。在此我想向所有参与其中的人表示感谢：Perry Arnold、Jon Becker、Tom Bogart、Karl Brown、Bryan Busse、

Katie Capps、Peter Chang、Scott Cohen、Stacey Doerr、Jeff Forbes、Stephen Freund、Jon Goldberg、Tague Griffith、Matt Harad、Lindsay Lack、Christine Lee、Tricia Lee、John Lilly、Albert Lin、Mara Mather、Hugh McGuire、Jeffrey Oldham、David O'Keefe、Bryan Rollins、Samir Saxena、Nikhyl Singhal、Eric Tucker、Garner Weng、Howard Wong-Toi和John Yong。

同时，我还要感谢所有使用本书并为本书的初稿提出意见的学生们：Adjo Amekudzi、Eric Anderson、Andrew Arnold、Kevin Berk、Kevin Carbajal、Ajit Chaudhari、Alida Cheung、Hye-won Choi、Elisabeth、Christensen、Ralph Davis、Joel Firehammer、Peter Frelinghuysen、Trevor Gattis、Teddy Harris、Heather Heal、Enoch Huang、Ann Lee、Ted Lee、Daphne Lichtensztajn、Lisa Maddock、Allan Marcus、Brad McGoran、Forrest Melton、Adrienne Osborn、Masatoshi Saito、Anne Stern、Ricardo Urena和Nichole Wilson。

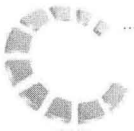
在1993～1994年，几位来自其他大学的教师试用了本书的初稿并提出了一些很有价值的建议。我要特别感谢哈佛大学的Margo Seltzer，内华达大学里诺分校的Rob Langsner，马里兰大学（巴尔的摩州）的Richard Chang，拉萨尔大学的Jane Turk和维斯特蒙德学院的Kim Kihlstrom。他们为本书早期阶段的修改提供了大力帮助。

我还要感谢对本书进行审校的人员：

| | |
|---|---|
| Stephen Allan | 犹他州立大学 |
| James Schmolze | 塔夫茨大学 |
| Don Goelman | Villanova大学 |
| Michael Skolnick | Rensselaer Polytechnic |
| Stan Kolasa | Rutgers大学 |
| Jeffrey A.Slomka | 西南得克萨斯州立大学 |
| Harry R.Lewis | 哈佛大学 |
| Kevin Smith | Emory大学 |
| Bill Muellner | Elmhurst学院 |
| Phil Tromovitch | SUNY-Stony Brook |
| Rayno Niemi | 罗切斯特理工学院 |
| John A.Trono | St.Michaels'学院 |
| Robert G. Plantz | Sonoma州立大学 |
| Robert Walker | Rensselaer PolyTechnic |
| David Rosenthal | Seton Hall |
| Richard Weinand | Wayne州立大学 |

此外，我的一些朋友和同事也对本书提供了有益的建议，他们是：Josh Barnes、Pavel Curtis、Kathleen Kells、James Finn和Steve Lewin-Berlin。

本书最终版本的成形还要感谢Addison-Wesley的编辑，他们在整个过程中一直不遗余力地提供着帮助。在这里要特别感谢Lynne Doran Cote、Sue Gleason、Peter Gordon、Laura Michaels、Jim Rigney、Karen Stone和Amy Willcutt所作的大量工作。我很幸运有Lauren Rusk作为我的合作伙伴和编辑，没有她这一切都不会成为现实。



# 目　录