国外经典教材

PEARSON Prentice Hall

Data structures and Program Design In C++

C++数据结构与程序设计

(美) Robert L. Kruse
Alexander J. Ryba 著
钱丽萍 译

清华大学出版社

国外经典教材

C++数据结构与程序设计

Robert L. Kruse
（美）　　著
Alexander J. Ryba

钱丽萍　　译

清华大学出版社
北京

内 容 简 介

这本精心制作的课本结合面向对象程序设计和 C++强有力的特性，构建数据结构的基本思想，设计了程序和有趣的应用。在此过程中，本书探讨了作为软件设计基本工具的问题求解和设计原理、数据抽象、递归和算法的比较分析。本书使用真实的案例研究、可重用的软件开发和程序设计项目来增强理解。

本书内容详尽且配有大量的实例和习题。书中所有算法都做了详细的注解，有利于读者理解算法的实质和编程思想。

本书既可作为高等学校计算机及相关专业学生的教材，亦可供计算机应用领域的工程技术人员参考，尤其适合于应用 C++语言编程的科技人员。

图书在版编目（CIP）数据
C++数据结构与程序设计/ （美）克鲁斯（Kruse, R. L.），（美）瑞贝（Ryba, A. J.）著；钱丽萍译. —北京：清华大学出版社，2003
（国外经典教材）
书名原文：Data Structures and Program Design In C++
ISBN 7-302-07804-1
Ⅰ.C… Ⅱ.①克… ②瑞… ③钱… Ⅲ.① C 语言—程序设计—教材 ② 数据结构—教材 Ⅳ. ① TP312 ② TP311.12
中国版本图书馆 CIP 数据核字（2003）第 116041 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

文稿编辑：汤涌涛
封面设计：久久度企划
印 装 者：清华大学印刷厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：37.5　**字数**：908 千字
版　　次：2004 年 1 月第 1 版　2004 年 1 月第 1 次印刷
书　　号：ISBN 7-302-07804-1/TP・5688
印　　数：1～4000
定　　价：59.00 元

读者对本书的赞誉

★★★★★

本书以前的 Pascal 版本特别易于阅读，也是我学习数据结构的用书。课本中包含的许多图表对于理解算法来说其价值是无法衡量的。本书对以前的版本做了进一步改进，加入了更直观的图片，并对旧书进行了更新，加入了旧书编写时未出现的一些新的算法和分析技术，如平摊分析。作者的编写风格完全易于理解。我收集有多本有关算法和数据结构的书，而这一本是迄今为止可读性最好的。

——Jeffrey Hsu，摘自亚马逊网站

★★★★★

我个人认为，作为数据结构的入门书籍，这本书是极好的。首先，它通过有趣而具体的例子成功地激发了学生学习相关主题的兴趣。其次，它注意使用图画和色彩，从而使得阅读尽可能容易。最后，作者在程序中精细地“提升”了“结构”的概念并在整篇课文中都说明了它的重要性。

——一名读者，摘自亚马逊网站

★★★★★

本书对那些需要研究构成通用程序的数据结构的 C 程序设计新手来说是优秀的资源。例如课文以一定的深度讨论了队列、栈、哈希表和递归，但这些讨论却是易于理解的。

——一名读者，摘自亚马逊网站

★★★★★

如果你是一个中级C++程序设计者，则本书以一种清晰的方式介绍了数据结构，书中有一些我在教科书中曾看到过的最好的图形。每一章的结尾都包含了非常好的练习和高级阅读物的列表(出色的特征)。也许对于高级学生来说，这本书显得不够详细，但我确实非常喜欢这本书。非常感谢作者！

——摘自亚马逊网站

前　言

木匠学徒可能仅仅需要一把斧头和一把锯子，而建筑师却使用许多精密的工具。计算机程序设计同样需要完善的工具来应对实际应用的复杂性，而只有不断使用这些工具进行实践，才能积累使用技能。本书将结构化问题求解、面向对象的程序设计、数据抽象以及算法的比较分析看作程序设计的基本工具。书中详细设计了几个相当规模的案例研究，以此说明如何同时使用所有这些工具来建立完整的程序。

我们所研究的许多算法和数据结构拥有内在的简洁性，这是一种掩盖了它们的应用范围和能力的简洁性。用不了多长时间，学生即发现在预备课程中经常使用的一些天真的方法得到了巨大的改进。然而方法的这种简洁性却被不确定性淡化了。学生很快又发现，几种方法中的哪一种在特定应用中能证明是最佳的，其答案远非显而易见。因此我们利用这个机会，及早地介绍那些既具有内在趣味性，又具有实际重要性的真正复杂的问题，并展示数学方法在算法验证和分析中的应用。

许多学生感到很难将抽象的概念转化为实践。因此，本书特别关注了将概念表达成算法和将算法细化成能运用于实际问题的具体程序。类似地，本书将数据说明和抽象的过程放在数据结构选择及其实现之前。

精心设计启发性的例子，继之以更通用的形式来表达这个概念，我们认为这是一个从具体到抽象的发展过程。大多数学生在其学业的早期，需要通过目睹所学概念的直接应用来加以巩固，他们需要编写并且运行程序，以阐明自己所学的每个重要概念。因此本书包含许多例子程序，有较短的函数，也有相当长的完整程序。另外，习题和程序设计项目构成了本书不可缺少的部分。其中许多习题和项目是所学主题的直接应用，经常要求编写和运行程序，以此可以测试和比较算法。有些是比较大的项目，并且有一些项目适合学生小组共同设计完成。

本书中的程序采用流行的面向对象语言C++编写。我们认为许多面向对象的技术为数据结构设计的基本原理提供了自然的实现。这样，C++允许我们构造安全、有效和简单的数据结构实现。我们承认C++很复杂，学生们需要通过学习一门数据结构课程来深化和提炼他们对这门语言的理解。在编写本书的过程中，我们尽量通过仔细介绍和解释C++各种面向对象的特征来支持这种深化。因此，我们在第1章开始时假设读者能自如地使用C++的基本部分(实质上是C的子集)，并逐步地加入C++的面向对象的元素，如类、方法、构造函数、继承、动态内存管理、析构函数、拷贝构造函数、重载函数和运算符、模板、虚函数和STL。当然，本书重点讲述的是数据结构，因此对不太了解C++的学生来说，阅读本书前需要先学习一本C++程序设计的课本。

大纲

* * * 程序设计原理

第1章通过完成首个大型项目(Conway的Life游戏)，详细说明面向对象的程序设计、

自顶向下的细化、复查和测试的原理，向学生展示这些原理，并且希望他们从头到尾都遵循这些原理。同时，这个项目为学生提供了一个机会来复习C++基本特征的语法，C++是本书通篇使用的程序设计语言。

＊＊＊栈

第2章介绍我们要学习的第一种数据结构——栈。本章将栈应用到反转输入、模拟桌面计算器和检查括弧嵌套等程序设计中。开始时利用STL栈实现，随后设计和使用我们自己的栈实现。第2章的一个主要目的是使学生理解信息隐藏、封装和数据抽象这些概念后面的思想，并将自顶向下的设计方法应用于数据和算法。本章最后对抽象数据类型进行了介绍。

＊＊＊队列

队列是第3章的中心论点。本章详细说明抽象数据类型的几种不同实现，并设计一个大型的应用程序来说明不同实现的相对优点。本章介绍了继承这一重要的面向对象技术。

＊＊＊链栈和队列

第4章提供栈和队列的链式实现。本章首先对C++中的指针和动态内存管理做一个全面的介绍。在展示一种简单的链栈实现后，我们讨论析构函数、拷贝构造函数和重载赋值运算符，它们都是链式结构的安全的C++实现中所需要的。

＊＊＊递归

第5章通过研究栈与递归问题的求解和程序设计之间的关系来继续阐明栈。本书通过研究几个实质性的递归应用来巩固这些概念，这些应用包括回溯和树结构的程序。如果需要的话，本章可以安排在此处之前、第2章结束后的任何时候学习。

＊＊＊表和字符串

第6章的主题是以链式和邻接实现的更通用的表。本章也包含封装的字符串的实现、C++模板介绍和对算法分析的非形式化的介绍。

＊＊＊查找

＊＊＊排序

＊＊＊表和信息检索

第7章、第8章和第9章分别介绍查找、排序和表格访问(包含散列)的算法。这些章节阐述算法和相关抽象数据类型、数据结构及实现之间的相互关系。课本中为初步的算法分析引入“大-O”和相关符号，并强调在空间、时间和程序设计强度的最佳利用方面作出关键的选择。这些选择要求我们找到分析方法来评价算法，而进行这些分析犹如一场战斗，需要利用组合数学这样的武器。在初级水平，我们既不期望学生全面掌握，也不期望他们拥有完备的数学知识而使自己的技能变得完美。因此，我们的目的是帮助学生认识到这些技能的重要性，期待以后有机会再学习数学。

＊＊＊二叉树

二叉树肯定是数据结构中最精致和有用的数据结构之一。第10章将对它进行研究，并将它与表、查找、排序中的概念相结合。作为递归定义的数据结构，二叉树为学生灵活地在数据结构和算法中应用递归提供了条件。本章以基本主题开始，并进展到诸如伸展树和平摊算法分析这样的高深的主题。

*** 多路树

第 11 章继续学习更复杂的数据结构，包括 Trie 树、B-树和红-黑树。

*** 图

第 12 章介绍图，将它作为对问题求解有用的更通用的结构，并介绍图中最短路径和最小生成树的一些经典算法。

*** 案例研究：波兰表示法

第 13 章中的案例研究相当详细地分析了波兰表示法，以此研究递归、树和栈作为问题求解和算法设计工具的相互关系。所涉及的一些问题能充当编译器设计的一个非形式化的介绍。我们照例将这个算法完全设计成一个能运行的C++程序，此程序接受输入普通（中缀）形式的表达式，将它翻译成后缀形式，并用指定的变量值评估此表达式。第 13 章可安排在 10.1 节完成后的任何时候学习。

附录中的几个论题并非切合于本书主题，但学生在课程准备过程中经常遗漏它们。

*** 数学方法

附录 A 介绍了离散数学中的一些论题。它的最后两节是斐波纳契数（Fibonacci number）和卡塔兰数（Catalan number），这两节更为复杂，并且并非课本中任何重要论题所必需的，但附录中将它们包含进来是鼓励更偏向数学的综合兴趣。

*** 随机数

附录 B 讨论伪随机数、生成器和应用，这是许多学生感兴趣的论题，但它不适合放在本课程的正文中。

*** 软件包和实用函数

附录 C 对本书中设计和多次使用的各种实用程序和数据结构软件包进行分类。附录 C 中讨论了声明和定义文件、变换单位、本书中通篇所用的实用程序包和一个计算 CPU 时间的软件包。

*** 程序设计技术规则、启示和易犯的错误

最后，附录 D 收集了所有的程序设计技术规则及分散在本书中的所有启示和易犯的错误，并将它们按主题组织以方便参考。

课程结构

*** 必备条件

学习本书的必备条件是学习过一门基本的程序设计课程，具有使用C++基本特征的经验。然而，因为我们仅仅是逐步谨慎地引入复杂的C++技术，我们相信，结合使用一本补充的C++课本和关于C++语言问题的更多的用法说明和重点，加之本书提供了C++版的数据结构教程，即使对于程序设计背景是另一门语言（如 C，Pascal 或 Java）的学生，这本书仍然适用。

中学数学的良好知识能满足几乎所有的算法分析的需要，但进一步的（可能是同步的）离散数学的知识准备将颇有价值。附录 A 中复习了所有要求的数学知识。

*** 内容

本书供诸如 ACM Course CS2（程序设计和实现）、ACM Course CS7（数据结构和算法

分析)或结合了这些课程的某门课程使用。本书完全覆盖了 ACM/IEEE 关于数据结构和算法的大多数知识单元①,它们包括:

AL1 基本数据结构,如数组、表、栈、队列、树和图;

AL2 抽象数据类型;

AL3 递归和递归算法;

AL4 使用大 Oh 符号的复杂性分析;

AL6 排序和查找;

AL8 带有大型案例研究的实际问题求解策略。

本书不论述 3 个最高级的知识单元:AL5(复杂性类,NP -完全问题)、AL7(可计算性和不可判定性)和 AL9(并行和分布式算法)。

本书中大多数章节都是首先介绍核心主题,然后给出例子、应用和更大的案例研究。因此,如果时间仅允许简要学习某个主题,则可以在不失连续性的情况下,仅学习核心主题,快速地从一章跳到另一章。然而当时间允许时,学习那些附加的主题和完成的项目都将使学生和教师从中获益。

*** 两学期课程

基本上可以用两学期的时间学完本书,学完本书后您就可以了解问题求解、数据结构、程序开发和算法分析等领域的许多论题。学生需要时间和实践来理解通用方法。通过将数据抽象、数据结构和算法的学习同它们在现实规模的项目中的实现相结合,一门综合的课程可以奠定一个坚实的基础,此后可以基于它学习更多的理论课程。即使本书没有完全覆盖它们,但也提供了足够的深度,使得感兴趣的学生在以后的工作中可以继续用它作为参考书目。无论如何,分配主要的程序设计项目并给予足够的时间来完成它们,这一点是重要的。

补充材料

除本书全部内容以外,还有一些补充材料,包括:

- 来自课本的所有的软件包、程序和其他C++代码段,其格式便于按需并入其他程序;
- 课本中一些示例程序和几乎所有程序设计项目的可执行的版本(DOS 或 Windows 平台);
- 课本中每节的提要或总结,适于用作学习指南。

这些材料可通过 ftp 方式获取:以 anonymous 用户登录进站点 *ftp. prenhall. com*,并转到目录 *pub/esm/computer_science. s-041 /kruse/cpp*。

致谢

这些年来,本书的 Pascal 版和 C 语言版得到许多人的帮助,他们是家人、朋友、同事和学生,在以前的书中曾特别提到过其中一些人。其他的许多人在学习这些书或是其各种语言

① 参见 Computing Curricula 1991: *Report of the ACM/IEEE-CS Joint Curriculum Task Force*, ACM Press, New York, 1990。

的译本的同时，友好地将他们的意见和建议转发给我们，所有这些都有助于我们将这本书做得更好。

感谢下列审阅者的建议，他们在许多方面帮助我们改进本书。他们是：Vander Linden（加尔文学院）、Jens Gregor（田纳西州大学）、Victor Berry（波士顿大学）、Jeffery Leon（伊利诺伊大学芝加哥分校）、Susan Hutt（密苏里-哥伦比亚大学）、Fred Harris（内华达大学）、Zhi-Li Zhang（明尼苏达大学）和 Andrew Sung（新墨西哥技术学院）。

Alex Ryba 特别感谢 Marquette 大学的 Wim Ruitenburg 和 John Simms，近年来从他们那里获得了富有成效的建议和鼓励，还要感谢以前的学生 Rick Vogel 和 Jun Wang 提出的意见。

Robert Kruse 特别感谢 PreTEX 公司 Paul Mailhot 不断提出忠告并给予帮助，他最初是一名出色的学生，后来作为一名可靠的研究助手，现在已成为一名得力的同事。他在书籍制作的软件开发，项目管理，对出版商、印刷工和作者的问题解答，以及为这项工作的各方面提出忠告和鼓励等方面都做出了重要的贡献。

没有 Prentice Hall 的编辑人员，特别是出版商 Alan Apt、采编 Laura Steele 和主编 Marcia Horton 热心的支持、真诚的鼓励和极大的耐心，这个项目不会开始，当然也不会完成。他们和其他制作人员的帮助是无价的。

Robert L. Kruse
Alexander J. Ryba

目　录

第1章　程序设计原理

本章首先概述良好程序设计的重要原理，特别是它们在大型项目中的应用；然后介绍用于发现有效算法的方法，如面向对象的设计和自顶向下的设计。在此过程中，我们提出将在后继章节中论述的程序设计和数据存储方法方面的问题，并通过使用C++编写程序，复习一下这门语言的一些基本特性。

1.1　简介

编写大型计算机程序的最大困难不在于确定此程序的目标是什么，也不在于找出达到此目标的方法。某个企业的总裁可能会说："让我们用一台计算机来记录所有的存货信息、账目报告和人事文件，并让它告诉我们何时需要再次订购存货，以及何时花费超出了预算线，并且还可以让它处理工资单。"花上充足的时间和精力，系统分析和程序设计员就可以确定企业的各个职员现在是如何完成这些任务的，然后就编写程序用同样的方式来完成这些工作。

＊＊＊大型程序的问题

然而，这种方法几乎必定是灾难性的失败。当系统分析员去调查职员时，他会发现某些工作可以轻易地放到计算机上去执行，于是他就这样做了。然后，当他们将其他工作移到计算机上时，往往发现它依赖于最初的那些任务。遗憾的是，这些任务的输出却或多或少地没有采用正确的格式，因此他们需要更多的程序设计，以将数据从一个任务给定的格式转换成另一个任务需要的格式。这个程序设计项目开始变得像是一床由各色布片拼缀成的被子，其中一些布片较牢，而另一些布片较弱；一些布片被仔细地与邻接的布片缝合，另一些却是勉强带上。如果程序设计员幸运的话，他们的作品可以足够好地连在一起，在大多数时间完成大多数的例行工作。但如果一旦必须做出任何修改，则整个系统将会有不可预测的结果。随后，又会出现新的需求，或者是一个出乎意外的问题，或许甚至是一个紧急事件，而这些程序设计员的成果所具有的功效就好比是用一张拼凑的被单作为从高楼跳下的人的安全网。

本书的主要目的是描述程序设计的方法和工具，这些方法和工具证明对现实规模的项目是有效的，这些程序比用来示范初级程序设计特征的那些普通程序大得多。由于将杂碎的方法用于解决大型问题注定会失败，因此我们首先必须采用一种一致的、统一的和逻辑的方法，也必须仔细遵守程序设计的重要原理。这些原理有时候在编写小程序时被忽略，但对大型项目，忽略它们将证明是灾难性的。

＊＊＊问题说明

处理大型问题时，首个主要障碍就是准确地判断这个问题是什么。有必要将模糊的目标、矛盾的要求和可能未明确说明的需求转换成能够进行编程的、精确规划的项目。人们以前使用的方法或对工作的划分未必最适于机器使用。因此我们的方法必须确定总体目标，

但却是准确的目标，然后渐渐地将工作划分成更小的问题，直到它们达到可管理的规模。

*** 程序设计

许多程序设计员奉行的准则是："首先让你的程序运行起来，然后使它变得优美。"这一准则对小型程序可能是有效的，但却不适用于大型程序。一个大型程序的每一部分都必须得到妥善地组织、清晰地书写和全面地理解，否则它的结构将被遗忘，也不再能够在以后某个时间连接到项目的其他部分，而那些部分可能由另一个程序设计员来完成。因此我们不把风格从程序设计的其他部分分离出来，但是从一开始我们就必须小心，养成良好的习惯。

*** 数据结构的选择

即使对非常大的项目，困难通常不在于未能找到一种解决方案，而是在于有太多可用的不同方法和算法，导致我们难以确定其中哪一个最佳、哪一个将导致程序设计困难或者哪一个可能效率低得让人绝望。算法设计中可变性的最大余地通常在于存储程序的数据的办法：

- 它们彼此如何被安排。
- 哪些数据保存在内存中。
- 哪些在需要时被计算。
- 哪些保存在文件中，这些文件又如何安排。

因此，本书的第二个目的是为数据组织和操作提供一些一流的但实质上简单的思想。表、栈和队列是我们最先学习的3种结构。随后，我们将为数据处理的重要任务设计一些强大的算法，如排序和查找。

*** 算法分析

当有若干不同的方法可以用来组织数据和设计算法时，开发一个用来推荐选择的标准就变得很重要了。因此我们将专心分析各种条件下的算法行为。

*** 测试和验证

调试程序的困难比程序规模的增长要快得多。即，如果一个程序的规模是另一个程序的两倍，则不一定花两倍的时间就能调试它，而很可能会是四倍的时间。许多大型程序（如操作系统）在交付使用时，仍然包含程序设计员无望找出的错误，因为这些困难看起来是难以克服的。有时候已耗时数年努力的项目由于无法发现为何不能工作而废弃。如果我们不希望自己的项目遭受这种命运的话，就必须使用满足以下性质的方法：

*** 程序正确性

- 减少错误数目，使得更易于发现那些剩余的错误。
- 能够预先验证我们的算法是正确的。
- 能够提供测试程序的方法，使得我们有理由确信这些程序不会行为失常。

这些方法的设计是我们的另一个目的，但它却并非完全在我们的能力可及的范围之内。

*** 维护

即使在一个程序开发完成、全部调试并交付使用后，仍然需要大量的工作来维护此程序的有效性。在对此程序有新的要求的时候，它的运行环境会改变，必须使之适合新的需求。基于此因，务必使一个大型项目编写得尽可能简单以利于理解和修改。

*** C++

程序设计语言C++是表达我们将面临的算法的一种十分便利的选择。Bjarne

Stroustrup 在 20 世纪 80 年代早期设计了这门语言，将它作为通俗 C 语言的一种扩展。Stroustrup 在C++中并入的大多数新特性有利于理解和实现数据结构。在C++最重要的特性中，用于我们学习数据结构的有：

* * * 要点

- C++允许数据抽象：这意味着程序设计员能创建新的类型来表达方便其应用程序的任何数据集合。
- C++支持面向对象的设计，在面向对象的设计中，程序设计员定义的类型在算法实现中起到中心作用。
- 重要的是，正如允许面向对象的方法，C++允许使用自顶向下的方法，这也是 C 程序设计员所熟悉的。
- C++便于代码重用及通用目的库的构造。这门语言包含了一个扩展的、有效的和方便的标准库。
- C++改进了 C 语言几个麻烦和危险的方面。
- C++保持了作为 C 语言特点的效率。

灵活性、通用性和高效性的结合，使得C++成为目前程序设计员最流行的选择之一。

我们将发现，C++的数据抽象和面向对象的特性自然地实现了作为数据结构设计基础的普遍原理。因此，我们将仔细解释如何使用C++的这些方面并当它们首次在本书中出现时简要地概述它们的语法(语法规则)。这样，我们将阐明和描述C++中许多不与 C 重叠的特性。关于C++语法的精确细节，请参考C++程序设计方面的课本——我们在本章最后的参考书目中推荐了几本这样的书。

1.2　Life 游戏

冒昧地引用一句古老的格言：实践出真知。

* * * 案例研究

贯穿本章，我们将集中学习一个案例。虽然按现实标准来看它并不大，但它阐明了程序设计的原理和我们应当学习避免的易犯的错误。有时候这个例子会带出通用的原理，有时候我们首先进行综合讨论，但我们的目的始终是发现能在实际应用范围中证明其价值的那些通用原理。在后继章节中，我们将对大型项目使用类似的方法。

我们将使用的例子称为 Life 游戏，由英国数学家 J. H. Conway 在 1970 年提出。

1.2.1　Life 游戏的规则

* * * 定义

Life 游戏实际上是一种模拟，并不是游戏者之间的游戏。它在一个无边界的矩形网格上进行，这个矩形网格中的每个单元可被一个有机体占据，或者不被占据。被占据的单元称为活的，未被占据的单元称为死的。哪一个单元是活的要根据其周围活的邻居单元的数目而一代代地发生变化，规则如下：