国外计算机科学教材系列

数据结构与算法

——面向对象的C++设计模式

Data Structures and Algorithms with Object-Oriented Design Patterns in C++

[美] Bruno R. Preiss 著

胡广斌 王崧 惠民 等译

WILEY

PHEI

電子工業出版社

Publishing House of Electronics Industry

www.phei.com.cn

内 容 提 要

本书是作者根据他在滑铁卢大学计算机工程学院教授数据结构与算法课程的经验编写而成的。它采用C++面向对象的设计模式，不仅系统全面地介绍了各种传统的数据结构，还把它们按照类和类层次的现代理念予以展开，进而达到抽象结构与实际设计的完美统一。本书的后三章通过引入抽象问题求解的概念，集中讲述了算法技术和各算法之间的关系。另外，作者运用一定的数学工具以及必要的分析技术和分析理论，对每种数据结构及相关算法都进行了时间和空间效率分析。

作为教科书，本书作者还在每章后面布置了习题和设计项目，并在全书的后面给出了问题参考答案，希望读者能从其中汲取宝贵的知识与经验。

WILEY

版权贸易合同登记号：01-1999-0928

图书在版编目（CIP）数据

数据结构与算法——面向对象的C++设计模式/（美）布莱斯（Preiss，B.R.）著；胡广斌等译. - 北京：电子工业出版社，2003.1

（国外计算机科学教材系列）

书名原文：Data Structures and Algorithms with Object-Oriented Design Patterns in C++

ISBN 7-5053-8339-6

Ⅰ. 数… Ⅱ. ①布… ②胡… Ⅲ ①数据结构—教材②算法分析—教材③C语言—程序设计—教材 Ⅳ. TP311.12

中国版本图书馆CIP数据核字（2002）第102837号

责任编辑：李　莹

印　　刷：北京天竺颖华印刷厂

出版发行：电子工业出版社　http //www phei.com cn

北京市海淀区万寿路173信箱　邮编：100036

北京市海淀区翠微东里甲2号　邮编：100036

经　　销：各地新华书店

开　　本：787×1092 1/16　印张：41.75　字数：1060 千字

版　　次：2003年1月第1版　2003年1月第1次印刷

定　　价：55.00元

凡购买电子工业出版社的图书，如有缺损问题，请向购买书店调换，若书店售缺，请与本社发行部联系。联系电话：（010）68279077

前　言

这本书是根据我教授E&CE250（滑铁卢大学计算机工程学的算法和数据结构）这个课程的经验写成的。我有幸目睹了面向对象的方法的出现以及它对数据结构和算法分析方面的教学产生的重大影响。这些技术的成功实施带来了一种认识上的一致性：当合适的设计模式和抽象方法被使用时，那些独立的明显不相关的思想好像得到了统一。

这种范例的转变是进化的和革命性的。一方面，随着程序员和研究人员不断发现新的算法和数据结构，这种知识的基础得到了长足的增长。另一方面，面向对象技术的正确使用又需要程序设计和实施方式上的根本变化。

目的

这本书的主要目的是为了促进用C++进行面向对象的设计并且阐述如何使用面向对象的设计模式。富有经验的面向对象的程序员发现了某些使工作完成得很好的方法并且不断地加以实践。这本书就是告诉你如何使用这些方式来实现好的软件设计。特别地，下面的这些设计方式贯穿于全文之中：单元素，容器，迭代器，适配器和访问者等。

事实上，所有的数据结构都在一个单一、统一、多态的类层次的关系中表现出来。这个框架清楚地表示了数据结构之间的关系并且阐述了多态性和继承性的有效应用。另外，当提到算法类时，算法抽象被广泛地使用。也就是说在描述一种一般性的算法时可以不用去关心它特殊的具体的实现细节。

这本书的第二个目的是提出一些数学工具，以及必要的分析技术和分析理论。在过去，当本书中的课程只向研究生教授时，作者依赖于学生已经有了在数学方面的基础理论。然而要将本书向大学二三年级的学生推介，就有必要加入一些所需的背景知识。我们的出发点是尽可能地培养他们对概念的直观感觉而不是对数学的严谨性。

方法

仅仅读一本书就学会编程序是可望而不可求的，经验必须要通过实践来积累并升华。然而，最好的实践就是研究他人的工作并把他们的观点融合到自己的实践中。我建议在学完基本的程序设计后，学生们应该接触一些复杂的、很好的程序，使他们能够了解如何设计好的软件。

为此，本书全部采用C++的程序片段介绍各种各样的数据结构和算法。所有这些程序段都是从正在使用的和经过测试的程序源代码中自动提取出来的。

ANSI标准C++语言的草案中提出的所有功能在这些例子中都被使用了，包括模板、例程和运行时类信息[3]等。根据我的经验，通过进行适当的提取，就有可能把这些概念应用于完全的功能性程序，而不需要求助于伪代码或手写。

概述

这本书中出现的素材在ACM/IEEE-CS联合课程任务的Computing Curricula 1991报告中已得到确认。本书特别专注于以下的几个知识单元：AL1：基本数据结构，AL2：抽象数据类型，AL3：递归算法，AL4：复杂性算法，AL6：排序和查找，以及AL8：问题求解策略。其覆盖的宽度和涉及的深度在计算科学/计算工程专业二或三年级学生的课程中最具有代表性。

为了分析一个程序，需要提出一个计算机的模型。在第2章中就有许多此类模型以及阐述这些模型如何预测性能的实例，它们都考虑了运行时正常的情况和最糟糕的情况。复杂性算法和如何用重复的代入法来解决重复性问题也是本章要讨论的，另外还有对数学、几何级数、Horner法则和调和数的特性等所做的总体回顾。

第3章介绍不对称（大-oh）符号，还通过与第2章比较说明不对称分析的结果和更高的逼真模型是相容的。除$O(\cdot)$之外，这章还讨论了别的不对称符号$\Omega(\cdot)$，$\Theta(\cdot)$和$o(\cdot)$及多项式和对数的性质。

第4章介绍基本数据结构——数组和链表。事实上，本书其余部分所有的数据结构都能用这些基本数据结构的一种或两种来实现。这章还要讨论多维数组和矩阵。

第5章介绍抽象的数据类型。它提出了贯穿于本书中所使用的重复的设计模式和后面几章中出现的一种统一的结构。特别是，所有的数据结构都被看作抽象的容器。

第6章讨论堆栈，队列和双端队列。本章提出了基于两种基本数据结构（数组和链表）的实现。堆栈，队列和双端队列的应用都有介绍。

第7章讨论顺序列表，包括有序和无序。在这一章里，列表被看作一个可查找的容器。关于列表的许多应用也有介绍。

第8章介绍散列和哈希表。本章致力于为各种基本数据类型以及第5章所介绍的抽象数据类型进行散列函数的设计，对分散表和哈希表都有相当深度的介绍并且得到了解析的性能结果。

第9章介绍树和它们的多种形态，以及树的深度优先和广度优先遍历。同时也基于访问者模式之上的完整的一般遍历算法，表明算法抽象的强大功效。本章还介绍如何用树表现数学表达式，以及遍历和各种各样的表达式符号（前缀，中缀，后缀）之间的关系。

第10章把树作为可查找的容器。通过揭示简单算法和平衡算法之间的关系，算法抽象的力量再一次得到了体现。本章还介绍平均概率性能分析以及通过伸缩性来解决循环等问题。

第11章介绍许多优先级队列的实现，包括二叉堆，左翼堆和二项式队列等。并特别阐述如何把一个比较复杂的数据结构（左翼堆）展开为已经存在的一种（树）。同时将离散事件的模拟作为优先队列的一个应用而提出。

第12章介绍集合和多元集合。还有分割和解体集合的算法，它再一次阐述了算法抽象。

第13章介绍动态存储管理，这在这一类的书籍中通常是找不到的。然而允许使用者重新定义new和delete操作符这一C++的特征使得这个主题具有可实现性。

第14章概述许多的算法设计技术。包括蛮干算法和贪心算法，回溯算法（包括分支界线法），分治算法和动态程序设计等。建立在一个抽象求解空间和抽象求解程序这两种概念

之上的面向对象的方法统一了许多的讨论。本章还简要介绍随机数的产生，蒙特卡洛方法和模拟退火等。

第15章介绍基于抽象排序概念的面向对象方式中的主要排序算法。另外，以算法抽象来阐述各种各样的排序算法类之间的关系并示范算法抽象的使用。

最后，第16章全面介绍图和图算法。包括深度优先和广度优先这两种图的遍历，以及被看作是另一种特殊遍历的拓扑排序。本章还提出基于访问者设计模式的一般的遍历算法，再一次阐述算法抽象，并对最短路径算法和最小跨度树算法做了介绍。

在每一章的结尾都有一套练习题和一套程序设计题：练习题用来巩固出现在文中的概念，程序设计则需要学生去拓展文中所给的实例。

建议课程大纲

本书可以在一学期或两学期的课程中使用。我在滑铁卢教这个课程时用的就是一个学期的36个学时，具体安排如下：

1. 复习C++程序设计的基础知识，概述用C++进行面向对象的程序设计（附录A）。[4个学时]
2. 计算模型，算法分析和渐近表示法（第2章和第3章）。[4个学时]
3. 基本数据结构，抽象和抽象数据类型（第4章和第5章）。[4个学时]
4. 堆栈，队列，线性表和排序表（第6章和第7章）。[3个学时]
5. 散列，哈希表和分散表（第8章）。[3个学时]
6. 树和查找树（第9章和第10章）。[6个学时]
7. 堆和优先队列（第11章）。[3个学时]
8. 算法设计技术（第14章）。[3个学时]
9. 排序算法和排序器（第15章）。[3个学时]
10. 图和图算法（第16章）。[3个学时]

取决于学生的背景知识，教师会发现重新回顾C++语言的特征是必要的。例如，在第4章中讨论的基本的数据结构需要对模板有一个理解。同样的，为了理解在第5章中讨论的统一类的层次，学生们需要理解类以及继承的工作。

联机课程资料

可以在下面的Web站点中找到支持本书的另外的一些资料。

http://www.pads.uwaterloo.ca/bruno.preiss/books/opus4

另外，你还会在那儿找到本书中所有程序片段的源代码以及勘误表。

Bruno R. Preiss
Waterloo, Canada
January 22, 1998

译 者 序

数据结构及算法分析是计算机专业的一门基础学科，在计算机各领域及各种应用软件中都要使用到。

本书的特点是：把数据结构的原理与算法分析技术有机地结合在一起，并对每种数据结构及其相关的算法都进行了时间、空间——主要是时间效率方面的分析，这就是算法分析技术。有时，对实现同一目的程序进行了多种算法设计，以分析它们在时间方面的差异。这在大型程序设计方面是经常要考虑的。另外，本书对数据结构及算法的描述，采用了当今广为使用的面向对象的C++语言——这使得本书的许多算法可直接应用于将来的实践中，从而增强了算法的移植性，就像你的“保留节目”一样。本书应用了C++中一些重要的特征，如继承、多态性、操作符重载、模板、对象与类等等。这些在本书的一些章节及后面的附录中作了一些说明，但是本书并不拘泥于C++的这些重要特征，也不是一本C++教材，而是从抽象的角度上讲述数据结构，使得算法的实现清晰易懂。

注意，本书广泛使用了一个非常重要概念——类。各种数据结构也是按照类和类层次展开的。首先，讲述抽象基类；再从抽象基类派生出具体类；然后，讲述类中的值集和操作集，即类的成员变量和成员函数。这些使得你很容易理解某一抽象数据结构是如何实现的。最后，在每一章的后面都举了一个应用例子，让你从中理解某种数据结构的应用，从而受到启发。

本书是按照从浅入深由一般到具体来讲述的。第2章算法分析及渐近性表示是算法分析技术的基础，在以后各章都要用到。第4章基本数据结构及第5章数据类型与抽象是后面数据结构及算法实现的基础。数组与链表是两种基本的数据结构，后面各种数据结构（从第6章到第12章及第16章的图）都用这两种数基本据结构来实现，并比较了它们在时间或空间上的效率。而随后的每一种数据结构都是一种ADT，它们的设计遵循着一定的规律——类层次，这也是面向对象设计的一种优点。第6章到第12章及第16章图的数据结构基本上可以分两类：一类为线性结构，包括栈及队列、有序线性表及排序表、哈希表和分散表等；另一类为非线性结构，包括树、查找树、堆及优先队列、集合及多重集和分区、图等。第15章排序历来是讲述的重点，同时也可以看到一种算法的优劣在时间上的差别，这从算法的运行时间对比上可以看到。第13章和第14章是高一级的内容，它们讲述在动态存储管理方面的应用及各种算法求解策略。采用何种存储管理，采用何种问题求解策略，这对计算机资源和运行效率都有非常重要的影响。这也是在大型程序设计时所要面对的问题。

最后是附录部分。附录介绍了一些必要的C++语法，从而使一些不懂C++语法的程序员能够更好地理解本书。

本书是由胡广斌，王崧等翻译的，限于水平难免有不妥之处，敬请批评指正。

目　录

第1章 概 要

1.1 本书的主要内容

本书的主要内容是数据结构和算法的基本原理——这是设计大而复杂的软件产品所必须具备的基础。为了彻底理解数据结构需要知道三件事：首先，必须了解信息在计算机里是如何安排的。其次，必须熟悉那些用来操作包含在数据结构中的信息的算法。最后，还必须理解数据结构的性能特征，使自己在为一个特殊的应用选择合适的数据结构时，能够作出恰当的决断。

本书还阐述了面向对象的设计，并且鼓励使用面向对象的设计模式。本书中的算法和数据结构全部是用C++来实现的。实际上，所有的数据结构都有一个单一的类层次相伴。这种约束使得后续章节中出现的程序能够建立于前叙章节中出现的程序之上。

1.2 面向对象的设计

传统的软件设计方法或是面向数据的，或是面向过程的。面向数据的方法强调信息的表示和全局之中各个局部之间的关系，而并不看重操作数据的行为。另一方面，面向过程的方法则强调一个软件执行的行为，却忽略了数据的重要性。

现在，众人一致认为在处理大而复杂的软件产品设计中产生的复杂性时，面向对象的设计方法比面向数据或面向过程的方法更为有效。这是因为数据和过程具有同等的重要性，使用对象可以把数据和操作数据的过程结合起来，从而充分发挥对象自身的主要优点——抽象性和封装性。

抽象性

抽象性可以理解为这样一种机制：在抑制不相关的细节的同时强调相关的细节。抽象性的一个重要的好处就是使程序员能更容易地解决问题。

例如，过程的抽象让程序设计者只需考虑所要执行的动作而不必担心这些动作的具体实现，数据的抽象让程序设计者只去考虑一个程序中的对象和对象之间的相互作用而不用担心如何实现它们。

抽象有许多不同的级别。低级别的抽象揭示了更多实现的细节，高级别的抽象则藏起了更多的细节。

封装性

封装性就是强制地隐藏信息，以此为软件设计人员提供帮助。对象把数据和操纵数据的过程封装在一起，从某种意义上讲，对于对象的使用者，对象隐瞒了实现的细节。

封装性有两个最现实的好处——概念上和物理上的独立性。概念上的独立性来自于对对象的使用者隐藏一个对象的实现从而使用户对一个依赖于该对象的实现的对象无法做任何事情。而这正是我们想要的，因为它允许不修改用户代码就可以改变实现的方法。

物理上的独立性来自于这样一个事实：一个对象的行为是由对象本身决定的，而不是由一些外在的东西决定的。因此，当我们在一个对象上执行一个操作时，不会有我们不想要的副面影响。

1.3 对象分级与设计方法

面向对象编程不只是对对象中的数据和操作这些数据的过程进行简单的封装，它还要对对象进行分类，并建立不同对象类之间的关系。

派生是表示对象类之间关系的最主要的手段——新类可以从已知类中派生。那么是什么使得派生如此有用呢？是继承。派生类继承了被派生类的特性。而且，被继承函数可以修改，另外的函数可在派生类中进行定义。

本书的一大特点是把所有的数据结构都按单一的类层次进行安排，而类层次就是关于数据结构的分类。对某一给定的抽象的数据结构有多种实现方法，但其每一种实现方法却能从同一基本类中派生。把一些类的共同特点抽象出来加以封装，又可依次派生出相关的基本类。

除了能分级处理相关的类之外，有经验的面向对象的程序设计者还可仔细考虑不相关类之间的相互作用。借助经验，一个好的设计者会发现对象之间相互作用重复出现的方式。通过学习使用这些方式，你的面向对象设计会更加灵活，可重复性更强。

最近，一些程序设计者开始给这些公用设计方法命名，而且，公用设计方法的目录编纂和出版工作也在进行之中。

以下这些面向对象的设计方法将一直贯穿全文。

容器

容器就是一个包含其它对象的对象，它具有一定的容量，可满，可空，而且可进行对象的嵌入和撤销。另外，一个可搜索的容器能支持高效率的查找操作。

迭代器

迭代器提供了一种手段，使得集合中的对象逐次逐个地得到访问。所有的迭代器共用一个公共接口，而集合的用户无需了解集合的基本实现方式。

访问器

一个访问器就是一项对容器中各个对象实施的操作。所有的访问器共用一个公共接口，因而隐藏了容器中的操作容器过程。同时，访问器可依据容器分别定义，即使是特殊的访问器也能为任何容器所使用。