普通高等教育“九五”国家教委重点教材

数据结构

——C++与面向对象的途径

张乃孝　裘宗燕

高等教育出版社

普通高等教育“九五”国家教委重点教材

数据结构

——C++与面向对象的途径

张乃孝　裘宗燕

高等教育出版社

（京）112号

内 容 提 要

本书采用面向对象的思想组织数据结构的内容，运用C++语言作为讨论数据结构的工作语言，系统地介绍了串，向量，链表，栈，队列，树，二叉树，优先队列，散列表，集合，字典，图和文件等各种数据结构，是一本比较好地反映了当前计算机技术发展水平，又具有较强实用性的数据结构教材。

全书体系完整，概念清楚，内容充实，取材适当，面向实际应用。可以作为大学本科计算机专业或相关专业的“数据结构”课程教材，也可以作为大学本科或专科有关专业的面向对象程序设计课程或C++程序设计实践课程的教材和参考书。

图书在版编目(CIP)数据

数据结构:C++与面向对象的途径/张乃孝,裘宗燕编著. —北京:高等教育出版社,1998
ISBN 7-04-006418-9

Ⅰ.数… Ⅱ.①张… ②裘… Ⅲ.①数据结构②C语言
Ⅳ.TP311.12

中国版本图书馆CIP数据核字(98)第12350号

高等教育出版社出版
北京沙滩后街55号
邮政编码:100009 传真:64014048 电话:64054588
新华书店总店北京发行所发行
国防工业出版社印刷厂印刷
*
开本 787×1092 1/16 印张 24 字数 590 000
1998年6月第1版 1998年6月第1次印刷
印数 0 001—5 148

定价 19.10元

前　言

数据结构是计算机科学教育中的一门核心课程，是一门理论和实际紧密结合的基础课，它讨论的是计算机科学技术领域里许多最基本的问题。随着计算机科学技术及其应用的飞速发展，计算机学科的许多领域都发生了根本性的变化，数据结构的教学内容和教学方法也需要不断更新。但是，无论发生了怎样的变化，数据结构课程在计算机科学教育中的重要地位和作用并没有改变。

数据结构也称信息结构。著名计算机科学家 P. Wegner 认为：计算机科学就是“一种关于信息结构转换的科学”。“结构”一词可以当作动词看待，解释为“把某些成分(或称原子、元素、成员等)按一定规律组织在一起的过程和方式”；也可以作为名词，理解为“把某些成分按一定方式组织起来而形成的实体”。对于复杂一点的程序，需要处理的数据往往很多，这些数据之间经常又有错综复杂的相互关系，要能够有效地存储这些数据及其相互关系，使之能够在程序中有效地使用，以及如何把它们组织起来，这是我们要解决的一个最重要的问题，数据结构课程讨论的就是数据的组织问题。实际上，数据结构问题不仅与程序设计有关，与计算机科学的许多其他领域也都有密切的关系。学好这门课程对于深入地理解计算机科学，继续学习计算机科学的其他课程是非常重要的。要搞好计算机方面的工作，无论是实际的应用开发工作，还是理论研究工作，本课程的内容都是必不可少的知识基础。

学习任何课程都必须了解它与本学科其他课程的联系，明确其在更广泛的学科领域整体中的位置，只有这样我们才能抓住学习的要领，理解课程内容的实质，对于数据结构也不例外。下面的图形形象地显示了数据结构在用计算机解决问题过程中的地位，以及它与计算机科学技术其他相关领域的关系。

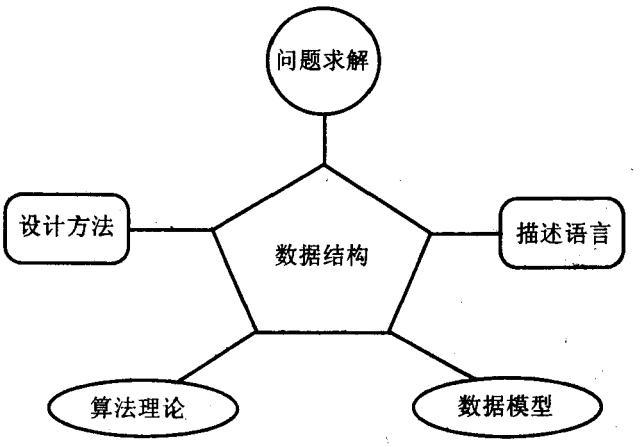

图 0.1　数据结构的地位及其与相关领域的联系

作为数据结构的基础，有算法分析与设计的理论和方法，以及关于数据模型的理论。实现数据结构需要采用某种描述语言(通常是某种程序设计语言)，并遵循一定的设计方法。学习和研究数据结构的目的是解决问题。

问题求解过程与数据结构

为了说明本课程的内容，我们看一个实际问题求解的例子。

用计算机解决问题常常可以看作对实际问题的一种模拟。从这种观点出发，工作目标就是在计算机中建立一个与实际问题有比较密切对应关系的模型。在这个模型中，计算机内部的数据表示了需要被处理的实际对象，包括其内在的性质和关系；处理这些数据的程序则模拟对象领域中的实际过程；最后，通过将计算机程序的运行结果在实际领域中给予解释，便得到了实际问题的解。

作为一个例子，考虑一个多叉路口(见图 0.2)交通管理系统的设计。在这个路口中，共有五条道路相交，其中 C 和 E 是单行线，其他为双行线。

通过分析，问题可以归结为对车辆的可能行驶方向作某种分组，每组包括一个或几个方向，对分组的要求是使任一个组中各个方向行驶的车辆可以同时安全行驶而不发生碰撞。显然对这个路口存在许多不同分组方案，而如果分组越少，可以同时行驶的车辆也就越多，可以认为这个管理系统的质量越高。

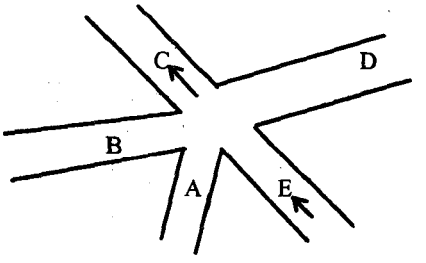

图 0.2　一个交叉路口的模型

根据这个路口的实际情况可以确定 13 个可能通行方向：A→B，A→C，A→D，B→A，B→C，B→D，D→A，D→B，D→C，E→A，E→B，E→C，E→D。有的方向明显不能同时进行，如 A→B 与 B→C 等。为了清楚和方便，我们把 A→B 简写成 AB，用一个结点表示(一个小椭圆)，在不能同时行驶的结点间画一条连线(表示它们互相冲突)，这样便得到图 0.3 所示的图式。这样得到的表示称为“图”。图是数学分支“图论”研究的对象，也是一种典型的数据结构。在本书的第十一章将讨论这种结构。

上面这样做的结果是把一个实际问题变成了一个抽象问题，即原问题的一个模型。问题求解的要求对应于将图 0.3 中的结点分组，使有线相连(互相冲突)的结点不在同一个组里。至此，对问题的分析完成了，要解决的问题借助图的模型(这是图论研究的数学对象)清楚而严格地表达出来。问题是：对于这个问题，能否设计一个总能够得到最佳的分组方案(人们

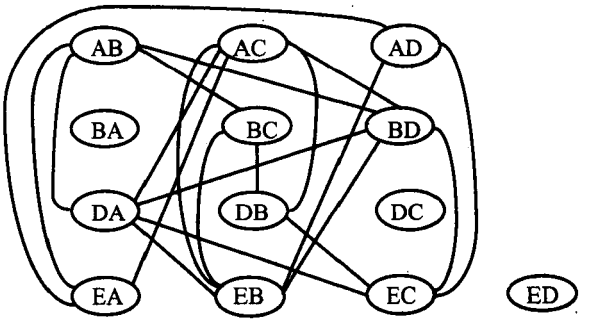

图 0.3　图 0.2 交叉路口的图式模型

把这种解称作最优解)的程序呢?

由于上面分析过程用了图的抽象表示方式,最后表达出的问题已经比最初提出的交通管理问题更一般、更抽象了,它反映了更大一类问题的要求。例如,如果把上图中的一个结点理解为一个国家,结点之间的连线看作两国有共同边界,上述问题就变成著名的"着色问题":即求出最少要几种颜色可将图中所有国家着色,使得任意两个相邻的国家颜色都不相同。对于一般的着色问题求最优解相当困难,至今还没有高效率的算法,可用的解法都是一个个穷举出所有可能的解,进行检查判断。这类穷举法对结点少的问题(称为"规模小的"问题)还可以用;对规模大的问题,由于求解时间会随实际问题规模的增长而指数性上升,这类方法实际上无法使用。因此人们对这类问题一般采用求近似最优解的方法处理。

求着色问题的近似解,一种常用方法的基本思想为:先用一种颜色给尽可能多的结点上色;然后用另一种颜色在未着色结点中给尽可能多的结点上色;如此反复直到所有结点都着色为止。选用一种新颜色给结点上色时要做以下工作:

1. 选出一个未着色的结点并用该新的颜色上色。

2. 寻找那些仍未着色的结点,如果某结点与用新颜色着色的结点没有边相连,则可将这个结点用该颜色上色。

把这种方法应用到关于交叉路口的例子中,可能得到如下一种分组(实际上,根据我们处理的顺序不同,可能得到不同的分组):

(1) 红色: AB AC AD BA DC ED
(2) 蓝色: BC BD EA
(3) 绿色: DA DB
(4) 白色: EB EC

这个解告诉我们,如果按这种方式对车辆行驶方向进行分组,就可以保证不会发生车辆相撞的事件。

下面的问题是如何将求解着色问题的这种想法在计算机上实现,也就是要写出一个程序,

对于任一个交叉路口的管理问题，都可以用这个程序给出解答[①]。为此需要做以下工作：首先，为问题中所有有关数据设计适当的表示形式，不仅包括需要表示的结点和连接，可能还有为计算过程的实现而用的辅助性数据结构。然后选择一种适当的程序设计语言实现这些数据结构，并在设计好的数据结构上精确地描述上面提出的计算过程，完成一个程序，使之能在计算机上运行。这些方面都是数据结构课程要涉及的。

对于这个具体问题，如果我们有一种很高级的语言，它可以直接描述和处理抽象的集合和图结构，那么程序的实现就非常简单。假设需要着色的图是 G，集合 V_1 包括图中所有未被着色的结点，着色开始时 V_1 是 G 所有结点集合。NEW 表示已确定可以用新颜色着色的结点集合。从 G 中找出可用新颜色着色的结点集的工作可以用下面的程序框架描述：

```
置 NEW 为空集合;
  for 每个 v ∈ V1 do
    if  v 与 NEW 中所有结点间都没有边
        从 V1 中去掉 v;  将 v 加入 NEW;
```

本程序片段每执行一遍，集合 NEW 中就得到可以用一新颜色着色的一组结点。着色程序的实现就是反复执行这个程序段，直到 V_1 为空。每次执行选择一种新颜色，程序段执行的次数就是需要的不同颜色个数。这个程序片段里涉及对集合和图的操作，包括由集合中去掉一个元素，向集合里增加一个元素，检查两个结点之间在图中是否有边连接等。有了这些结构和操作，程序的实现非常简单。

但是，常见程序设计语言并不直接支持图和集合等数据结构，通常只提供了一些基本数据类型(例如整数、实数、字符等)和一些数据构造手段(例如数组、记录、指针等)。要解决这个计算问题，我们必须自己用选定的语言所提供的机制实现这些结构(集合、图等)和操作(对集合的元素增删、对图的边的判断等)，这些正是本书将要讨论的基本内容。实际上，这个问题具有普遍性，在计算机问题求解过程中，人们往往需要设计适当的数据结构，并设计有关的算法，然后用一种程序语言实现所需要的结构，实现所需要的程序。数据结构问题在这个过程中的重要性是非常明显的。

各种典型的数据结构

许多实际计算机应用中都要用某种方式存放一组相关的数据对象，经过几十年的研究和实际工作，人们提出了许多数据组织方法，提出了许多不同的数据结构。为了学习和研究的方便，计算机科学家把常用的数据结构按照它们的特点进行整理、分类，并对各种结构的特性做了许多研究，总结出许多典型的数据结构。本书的主要内容就是讨论这些典型的数据结构，包括它们的结构、性质、实现方法和应用。这里，我们首先对本书中将要讨论的各种主要结构做一简单介绍：

① 实际上，这个程序的应用范围还更广泛，只要一个问题可以化为着色问题，就可以用这个程序求解。

“向量”是一种简单结构。元素在向量中顺序存放，每个元素排定一个编号，称为“下标”或“指标”。向量适合表示那种元素个数已知的或者至少能够确定其数量范围的对象。向量的另一个优点是所有元素都可以通过下标直接访问。

“字符串”是以字符作为元素的向量，字符串成员(字符)也可以直接访问。字符串的特点是每个成员只占很少的存储空间。字符串的应用广泛，如何有效地实现字符串是考虑的重点问题。

“表”是另一种顺序性数据结构。表中可以容纳的元素通常没有数量限制，因此它适于表示事先无法确定数目的元素序列。对表元素的使用通常只能从表的一端(称为表头)开始顺序地逐个进行。表的第一个元素称为表头元素，最后一个元素称为表尾元素。虽然表中的每个元素在表里有一个顺序排列位置，但对它们不能由位置出发进行直接访问。

“堆栈”和“队列”是两种常用的辅助存储结构，可以向其中存入元素，或从中取出元素。堆栈元素的存储和访问按照后进先出(Last In First Out， LIFO)原则，最先取出的总是在此之前最后放进去的那个元素。队列实现先进先出(First In First Out， FIFO)的访问原则，最先到达的元素也最先离开队列。堆栈和队列都是非常重要的数据结构，在计算机领域的各个方面应用非常广泛。

“树”和“二叉树”与前面几种结构不同。向量、表等都具有线性的结构，而树和二叉树的结构比较复杂。树的元素通常称为“结点”，树表示了结点的层次性关系。树最上面一层只有一个元素，称为“树根”。每个上层元素可以有若干相关联的下层元素，这些元素被称为是该上层元素的“子结点”；每个下层元素至多有一个对应的上层元素，称为它的“父结点”。许多实际的和理论的问题中都可以抽象出某种树形结构来。

“集合”是数学概念,也是一种基本数据结构。集合数据结构的特点在于许多运算是对整个数据结构进行的，其基本运算包括求并、求交、求差集、子集、和相等性检测等。集合是与顺序无关的元素的汇集，其中的每个元素都是唯一的。

“散列表”可以看作一种广义的向量，元素的访问同样通过下标进行。散列表的特点是不要求元素下标必须是整数，允许以任何类型的数据作为下标。散列表通过一个函数把下标值映射到整数值，然后再进行直接存取，这种映射函数被称为“散列函数”。人们经常把散列表的元素用其他结构实现，这样可以更好地发挥散列表的作用。在这种情况下，散列表的每个项可以看作是一个存储桶，其中能够存放多个数据值。如何实现存储桶是一个值得进一步考虑的问题。

“字典”(又称为“映射”)也是一种具有下标性质的结构。字典下标不像向量那样仅限于整数，允许采用任何类型的下标。字典下标值本身也需要存储在结构里。这样，字典可以看作是一种关联的集合，每个关联包含着一个下标(或称关键码)和一个值。所以抽象地看，一个字典就是由关键码集合到值集合的一个映射。字典在计算机领域应用广泛，人们提出了这种结构的许多实现方法。

“图”是一种较复杂的结构，它包括一个结点集合和一个边集合，边集合中每条边联系着两个结点。信息可以存储在结点里，也可以存储在边里。许多实际问题中的数据可以用图表示，如公路网络、通信网络、不同事物间的联系，等等。

本书中还介绍了另外一些重要数据结构，下表中列出了书中讨论的各种主要结构，并简单地总结比较了它们的特点。

结　构	主 要 特 性	性质方面的限制
向量	元素的直接访问	不适合大量的动态操作
表	线性结构，适合于动态操作	顺序访问，只能直接访问首元素
堆栈	LIFO 规则	只能访问最后放入的元素
队列	FIFO 规则	只能访问现存的最早放入元素
树	表示元素间的层次结构关系	算法比较复杂
AVL 树	保证元素访问的 log 操作时间	保持平衡的算法比较复杂
优先队列	对最小元素的快速访问	每次只能访问一个最小元素
散列表	实现对分散存储的快速访问	需要考虑合适的散列函数
集合	求数据集合交、并等操作	元素无序
字典	关键码与值的关联	结构复杂
图	表示数据间复杂的相互联系	结构复杂，无序
文件	存储在外存的结构	访问速度慢

抽象数据类型

本书中将采用抽象数据类型(Abstract Data Type 简称为 ADT)的观点讨论各种数据结构。抽象数据类型是人们在研究数据组织时提出的一种观点，一个 ADT 不仅定义了有关数据的一种组织方式，还包括了与这些数据有关的基本处理方法。ADT 可以看作是定义了相关操作的一个数学模型。例如，集合与集合的并、交、差运算就可以定义为一个典型的抽象数据类型。一个抽象数据类型要包括哪些操作，这一点由设计者根据需要确定。例如，对于集合，如果需要，也可以把判别一个集合是否空集或两个集合是否相等作为集合上的操作。

采用抽象数据类型的观点和方法实现数据结构，最重要的一个结果是可使数据结构变成一种“抽象”的东西。数据结构的具体实现方法在外面看不见了，能够看到的只是：有这样一个数据类型，它具有我们所期望的数据结构的性质，提供了我们希望这种数据结构所需要的各种操作。通过这个类型我们可以定义实际需要的数据结构对象，通过这些操作我们可以使用这些数据结构对象。

这种方式带来一些非常重要的特性：使数据结构的实现和对它的使用分离，使我们能够完全独立地考虑数据结构的实现问题，以及如何使用已经定义好的数据结构实现所要求的计算过程。这符合人们倡导的“模块化”这种实现复杂软件的基本方法。实现和使用的分离也使我们的程序不依赖于数据结构的具体实现方式。只要提供了相同的操作，换用一种其他方式实现的同样数据结构后，使用该数据结构的程序完全不需要修改，这个特征对系统的维护和修改非常有利。

在许多程序设计语言中预定义的各种类型，例如整数类型、浮点类型、指针类型等，都可以看作是简单的抽象数据类型。这些类型的元素就是程序中经常使用的数据(常量或变量)，这些数据的内部表示和操作的具体实现都是系统提供的，作为用户不必知道。用户要了解的只是与各种类型有关的操作的用法，比如说，说明一个整数的方式和对整数能执行的各种运算(+、－、×、/ 和取余等)。正是高级程序语言的这类特征使得我们在一种机器上实现的程序能比较方便地搬到另一种计算机上。可以说，抽象数据类型的观点是对这些的推广。

从本质上讲，数据结构课程和书籍中讨论的向量，表、栈、队列、二叉树、集合及字典等，也可以像普通的整数、浮点数等一样定义为程序设计语言层次上的抽象数据类型，有些新的语言也在考虑把这些数据结构用标准程序库的方式提供出来。问题是数据结构中讨论的这些抽象数据类型比较复杂，而要定义和实现的实际需求变化比较多，因此要在语言层次上提供能满足所有人需要的数据类型，其难度是非常大的。因此，这些结构往往需要由从事程序设计的人自己设计和实现。

要实现一个抽象数据类型，一要确定该类型内部的数据组织方式；二要定义提供给外部使用这种类型的界面描述。包括该抽象数据类型的名称，有关数据组织的描述和操作的描述等。作为一个例子，我们看一个“圆”数据类型的描述。通常圆(circle)定义为到一定点的距离等于定长的所有点的集合。一个一般的圆的抽象数据类型对象，其数据成分应该包括圆心和半径(radius)。如果我们只考虑圆的大小，那么在这个抽象数据类型中就只需要有半径数据。假设我们要设计一个 Circle 抽象数据类型，它包括计算面积(area)和周长(circumference)的操作，下面是 Circle 的一个描述(这里没有给出计算圆周长和面积的操作的实现)：

ADT circle is
  data
    非负实数作为圆的半径
  operations
    constructor
      输入的初值： 圆的半径
      处理： 赋半径的初值
    area
      输入： 无
      输出： 圆的面积
      处理： 计算圆的面积
    circumference
      输入： 无
      输出： 圆的周长
      处理： 计算圆的周长
end ADT circle

这个例子非常简单，而许多实际需要的数据类型可能要复杂得多。对于这样一种抽象数据类型的说明，我们可以用不同的数据表示方式和不同的算法实现，得到的结果在应用方面就可能有许多不同。关于这方面的讨论将是本书的一个重要内容，而评价这些不同实现的最重要指标是，实现数据结构所花费的空间代价以及实现操作所花费的时间代价和空间代价。

程序设计方法

计算机学科是一个年轻而发展十分迅速的学科。就程序设计方法而言，几乎每 10 年就有

一个大飞跃。20世纪50年代程序设计主要使用机器语言和汇编语言，程序员的工作像手工绘画绣花，没有固定方法，程序设计主要依赖于程序员的经验和习惯。

60年代流行的“面向过程”的方法，以问题的处理过程为中心，根据求解问题的算法，选择数据的表示和处理，然后编制出程序。ALGOL、FORTRAN和COBOL语言就是当时广为使用的典型面向过程的程序设计语言。

70年代“结构程序设计”思想被计算机界广泛接受和使用，程序员的工作有了更多的科学依据。采用结构程序设计方法提高了程序可读性和可维护性，其目的不仅在于编制出结构良好的程序，还能够使程序设计过程进一步规范化。结构良好的程序使程序的意义更容易把握。开发这种结构良好程序的主要方法是“自顶向下”、“逐步求精”以及“模块化”。支持结构程序设计的代表性语言为：Pascal、Modula-2和Ada等。结构程序设计是程序设计方法的一大飞跃。

“面向对象的程序设计”(object-oriented programming)在80年代兴起。遵循这种途径，被处理对象的内部实现和它的外部联系被严格地区分开来。这样，既把对象的表示(数据结构)和行为(处理算法)封装起来，得到了很好的保护，又为对象之间的联系提供了必要的描述和处理手段。在面向对象程序设计中，“对象”(object)和“消息”(message)是两个重要的概念，它们分别用于表现事物本身的性质和事物间的联系。对象的行为用被称为“方法”(method)的程序段定义。一般地，程序中的对象被划分为一些“类”(class)，属于一个类的所有对象都具有同样的结构和行为。通过给一个对象发送消息来触发需要的行为。一个类还可以继承(inheritance)其他类的数据成分和用方法定义的行为。

面向对象的设计方法实际上是围绕组成问题领域的各种事物进行程序设计，关心的重点在于客观对象及其相互间的联系。用这种方法开发出来的系统，其内部由许多活动的对象构成，对象间的联系只通过消息传递进行。这样，在客观问题与解决问题的程序之间存在一种比较自然的联系，提高了程序的易理解性、易修改性和易维护性。继承机制又为重用原有代码和设计提供了有力的手段，加上后面我们还要介绍的多态性(polymorphism)思想，这些都可以提高程序的简明性、灵活性和可重用性，为设计和实现复杂的大程序提供了强有力的支持。面向对象的思想起源于Simula-67语言中类的概念和Smalltalk语言的对象概念，目前使用最广泛的面向对象程序设计语言是C++。本书将用C++作为描述语言，采用面向对象的方法讨论数据结构的实现问题。

本书的特点和结构

本书是一本系统介绍数据结构的教材。在本书的论述过程中会反复联系到计算机领域的许多重要问题，包括问题求解、算法的分析与设计、抽象数据类型、程序设计方法、程序设计语言等方面的内容。根据目前计算机科学发展的情况，本书采用面向对象的思想组织与这个课程有关的内容，运用C++语言作为讨论数据结构的工作语言，其目的是为了保证本书的先进性和适用性。

正如前面讨论的那样，数据结构本身是一些抽象的结构，并不依赖于实现它所使用的语言。即使这样，使用什么语言讨论数据结构问题仍然非常重要。选择一种实际程序语言作为课程中的工作语言，有利于使我们的讨论更面向实际应用，使参加课程学习的学生不仅可以

'掌握数据结构的抽象概念和性质，也可以了解如何实现以及实现中可能遇到的各种问题。由于程序设计语言的发展，适合用来讨论数据结构有关内容的语言越来越多，今天世界上新的数据结构教材已经不再用伪语言作为工具了，这是一个大趋势。从这方面看，采用一种应用广泛的程序语言是合适的。

采用面向对象的方法讨论数据结构还有一些优点。在本书中，我们不仅讨论了抽象的数据结构，而且讨论了各种数据结构的不同实现方式并对它们进行了比较，从中可以看到不同数据结构的相互关系，包括结构上的关系和实现关系。通过一学期的学习，学生不仅能够学到数据结构的基本知识，同时还能学会根据面向对象的思想，利用课本中已有的基本数据结构，构造新的面向应用的数据结构的方法，提高用计算机解决实际问题的能力。

由于 C++ 语言的一些优点，特别是它的模板机制，使我们可以比较清晰地实现具有通用性的数据结构。例如，我们实现的不是整数的堆栈，也不是字符串的堆栈，而是一般的堆栈。一个实现可以满足各种程序对于堆栈的需要。这样实现的数据结构实际上更接近那种抽象的概念上的数据结构。

综上所述，本书的讨论力图反映计算机科学的最新发展，用面向对象的方法组织数据结构的主要内容，以抽象数据类型的方式定义各种结构的用户界面，以时间和空间开销作为评价各种结构实现方法的主要标准，从简到繁，系统地介绍各种常用的数据结构，强调不同结构之间的联系。书中采用 C++ 语言描述各种类的界面和实现。把面向对象的程序设计方法与数据结构的主要原理紧密结合起来，讲解过程中还插入大量实例。读者通过本书的学习不仅能掌握数据结构的基本原理和基本方法，同时可以把所学的知识用于实际问题求解的过程之中，是一本实用的数据结构教材。

本书假定读者有基本的计算机基础和使用知识，具有基本的程序设计经验，掌握了 C 语言程序设计的基本技能(例如，经过一个学期 C 语言程序设计课程的学习)。本书适合作为大学本科计算机专业或相关专业的“数据结构”课程教材，或其他专业的计算机提高课程教材，也可以作为大学本科或专科有关专业的面向对象程序设计课程的教材，或程序设计实践课程的教材和参考书。大学专科计算机专业的“数据结构”课程也可以通过选择一些基本章节的方式使用本教材。对于超出教学大纲的或难度较大的章节，书中已用星号标出，供教学时参考。

Borland 公司开发的 Borland C++程序设计系统是在 Turbo C++和 Turbo C 程序设计系统基础上发展而来的。Borland C++2.0 包含了 AT&T C++语言 2.0 版本的全部功能，C++模块与 C 模块兼容，可以任意连接，同时增强了模块间设计接口的能力，支持面向对象程序设计的完整功能。本书的教学过程可以使用 Borland C++作为程序设计实验的基础系统，也可以使用其他支持 AT&T C++语言 2.0 版本的 C++语言系统。

全书共分十二章：

第一章，预备知识。主要介绍面向对象程序设计(OOP)的一些重要概念和 C++ 语言的基本特征。通过本章的学习，读者能了解：什么是抽象数据类型；什么是类；什么是对象；C++ 中的类与抽象数据类型的关系；面向对象的程序设计等。此外，在这一章的最后还讨论了算法设计与分析的一些基本问题。

第二章，字符串——数据封装技术。本章定义了一种更安全可靠的字符串类型，解决了

C/C++ 语言的一个缺陷。同时也以字符串作例子，讨论数据抽象和封装的有关问题，介绍 C++ 语言中这方面的特征，以及这些特征的使用方法。

第三章，向量——程序重用技术。本章建立了一种安全可靠的向量数据类型，并为它实现了许多有用的操作，还给出了几个主要的向量排序算法。通过这一章的学习，读者应该掌握通用程序模块的实现技术和不同途径的重用技术。

第四章，遍历器——继承和多态性。本章讨论与继承和多态有关的问题。遍历器是一种非常一般的类，用于构造针对复杂结构类型的循环。向量遍历器的导出是继承方法的典型应用。

第五章，动态数据结构——链表。主要讨论了各种常用的链表结构及其实现方法。链表和向量是最基本的数据结构，是后面许多复杂数据结构的实现基础。

第六章，栈和队列。介绍了栈和队列的抽象概念、具体实现及其应用。

第七章，树和二叉树。介绍了树和二叉树的概念，重点介绍二叉树的实现，及树结构用于快速检索的一些技术。

第八章，优先队列。优先队列用于支持快速查找和删除集合中最小元素，它可以用表、树、向量等结构实现。本章主要介绍了堆和斜堆的概念，以及通过它们实现优先队列的方法。

第九章，散列表。本章介绍散列表及其实现。特别讨论了不同的关键码具有相同的散列值(碰撞)的处理问题。

第十章，集合和字典。本章介绍了集合和字典的概念，以及实现它们的各种实用技术。

第十一章，图。这一章里讨论图的概念，并且考虑这个概念的几种不同实现方式，介绍了图上的一些重要算法及其应用。

第十二章，文件。介绍文件的概念，包括流文件及其操作，把文件作为数据结构实现机制的方法，用文件实现字典的一些问题，特别是索引结构等。

感谢

这本书的形成包含着许多人的思考和贡献。作者曾用本书的初稿在北京大学校内讲授过数据结构课程，数学学院九五和九六级的同学和数学学院参加课程辅导的研究生们就这个课程和本书的初稿提出了许多宝贵意见。在本书的修改和整理过程中刘海燕同学做了大量输入和整理工作。作者特别感谢北京航空航天大学麦中凡教授和北京大学韩玉真教授，他们认真地审阅了全部书稿，并提出了许多意见和建议。

由于面向对象是一个比较新的领域，在如何用面向对象的观点讲授数据结构方面作者的经验不足，需要进一步思考和探索。因此在本书中难免会有一些错误和不足之处，恳请读者批评指正。

张乃孝　裘宗燕

1998 年 2 月于北京大学

目　录