数据结构与算法(C++)

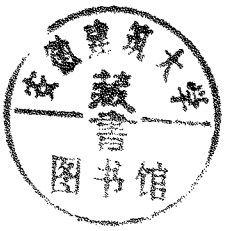
窦延平 张同珍 姜丽红 陈玉泉 编著

上海交通大学出版社

上海交通大学重点教材

数据结构与算法(C++)

窦延平　张同珍　姜丽红　陈玉泉　编著



上海交通大学出版社

内 容 简 介

“数据结构与算法”是一门重要的基础理论课程。它不但是计算机科学技术专业的核心课，同时已经成为理工类学生的一门必修课。本书运用面向对象的方法和C ++ 语言讲述数据结构与算法中的基本理论，并从抽象数据类型 ADT 的设计、表示和实现，C ++ 支持数据抽象、过程抽象、支持类属数据结构的手段统一描述各种数据结构与算法，使得各种常用的数据结构，如堆栈、队列、各种线性表、树、图、排序、查找、队列、优先队列更加条理和系统化。除此之外，本书从面向对象的角度讨论了算法设计的基本方法，做到了从面向对象和面向过程两个方面，在基本理论和基本技能上对学生进行强化训练。在本书最后一章，从应用的角度讨论了标准模板库 STL，把最新的支持数据结构与算法的手段介绍给读者。

本书内容丰富、深入浅出，适合于计算机类、电类、信息类、数学类、自动控制类学生作为教学用书；经过适当的选择，同样适合其他理工类学生作为教材使用；对于软件设计人员、工程技术人员也具有一定的参考价值。

图书在版编目（CIP）数据

数据结构与算法：C++/窦延平等编著. —上海：上海交通大学出版社，2005
ISBN 7-313-03943-3

Ⅰ.数… Ⅱ.窦… Ⅲ.①数据结构②算法分析③C语言－程序设计 Ⅳ.TP311.12

中国版本图书馆CIP数据核字（2005）第003262号

数据结构与算法(C++)

窦延平 张同珍 姜丽红 陈玉泉 **编著**

上海交通大学出版社出版发行

（上海市番禺路877号 邮政编码200030）

电话：64071208 出版人：张天蔚

常熟市文化印刷有限公司印刷 全国新华书店经销

开本：787mm × 1092mm 1/16 印张：21.5 字数：530千字

2005年5月第1版 2005年5月第1次印刷

印数：1—5 050

ISBN7－313－03943－3/TP·606 定价：32.00元

序　言

“数据结构与算法”是一门重要的基础理论课程，它不但是计算机科学技术专业的核心课，而且由于随着计算机技术向其他领域的广泛渗透，这门课程已经成为理工类学生的一门必修课。有鉴于此，上海交通大学电子信息和电气工程学院把本门课程列为学院大平台课程的一门基础技术课，使得无论是学习计算机，还是学习电子通信、电力、自动控制、仪器的学生，从大学一年级起，就打下运用面向对象的程序设计技术和应用、选择、评价各种数据结构与算法的牢固的基础，为大学生们拓宽计算机程序设计的应用领域、灵活运用数据结构与算法的基本理论，并同以后所要学习的专业灵活地结合起来，提供了有力的保障。

运用面向对象的技术和方法讨论数据结构与算法中的基本理论，是近年来的一股新的潮流。面向对象的技术和方法，对于抽象数据类型(Abstract Data Type)的设计、表示和实现，对于数据抽象、过程抽象、类属数据结构(Generic Data Structures)的表示和实现，都有强大的支持手段和独特的优点。因此，本书选用了C ++ 语言描述数据结构与算法。最近几年，标准模板库(Standard Template Library)技术越来越成熟，成为支持数据结构与算法的重要工具，因此本书在数据结构与算法的基本理论介绍之后，从应用的角度讨论了标准模板库 STL，支持 STL 的主要技术手段——类、模板、多态、继承和虚函数等等。STL 的重要优点之一，是把程序员从实现许多标准的数据结构与算法的令人厌烦的工作中解脱出来，因此可以大大加快程序设计的进程，使程序设计更加模块化、通用化、标准化。STL 由许许多多的组件组成，目的是为程序员提供一整套基础软件“芯片”。在掌握了基本数据结构与算法和C ++ 语言之后，理解 STL 的设计思想和实现方法，并不是一件困难的事。因此，本书把 STL 专门作为一章，建议大家在掌握了基本数据结构的理论之后，就可以研究一下这些数据结构在 STL 中是如何实现的，从而把两者有机地结合起来。这样做的好处是不会冲淡基本数据结构的学习，但又能很快地掌握标准模板库 STL。

本书第 1 章阐述什么是抽象数据类型 ADT，ADT 的表示和实现，构造数据结构的手段和过程。介绍了评价数据结构好坏的主要标准，算法的时间复杂性和空间复杂性，并引入了评价算法的几个符号，即大 O、大 Θ、大 Ω 和小 o 表示符号。

第 2 章到第 4 章介绍基本数据结构线性表、堆栈、队列的表示和实现及其应用。主要是针对C ++ 的特点，着重描述如何用类、继承等手段以及如何用动态存储结构表示这些常用的数据结构。在第 4 章中重点介绍用于模式匹配的 KMP 算法。

第 5 章讨论树和二叉树，它们的存储表示、特性、遍历和实现，并介绍了在理论和实践上都很重要的线索树、最优二叉树的分析和 HUFFUMAN 算法等。

第 6 章讨论各种查找技术。如有序表的查找、折半查找法、排序二叉树、HASHING 查找技术等，还讨论了 AVL 树、红黑树、顺序统计、B - 树及 B + 树，并分析了各种查找算法的优劣、算法复杂性的级别等。

第 7 章介绍图的存储、遍历的基本方法，并介绍了图的常用算法及其实现，如最小生成树、最短路径、拓扑排序、关键路径的求法等，并分析了图的算法的时空复杂性等。

第8章介绍各种排序方法,如快速排序、堆排序、合并排序、基数排序等,并给出了各种排序方法的时间复杂性分析。

第9章是基本算法设计,介绍了分治法、回溯法、动态规划算法、分支限界法,并给出了它们的形式描述、效率分析及其实现方法。

第10章是标准模板库,介绍标准模板库的构成和功能,介绍了各种容器、迭代器、算法和函数对象的使用和实现,有助于程序员摆脱各种常用数据结构与算法的编写,加快程序设计的速度。

本书虽然是针对电子信息和电气工程学院的学生编写的,但是对于大部分理工类学生都是适合的,同样也适合软件设计人员、工程技术人员使用和参考。根据学时和重点的不同,可以有选择地学习其中的一部分。在具体的教学安排上,应该做到削枝强干、突出重点。对于基本的数据结构的理论和方法必须做到精讲多练,勤于实践。根据我们多年的教学经验,本课程是一门理论性和实践性都很强的课程,必须做到理论和实践并重。上机练习中的程序调试,是从失败中学习提高的过程,对于加深理解、掌握本课程的精髓是必不可少的。

参加本书编写的教师都是计算机系的从事数据结构和算法教学多年的第一线的教师,具有丰富的教学经验。本书中所有的程序都在微软公司的VC++6.0编译程序上进行了调试和验证。本书的第1,7,9,10章由窦延平老师编写,第2,3,4章由张同珍老师编写的,姜丽红老师编写了第6章和第8章并和陈玉泉老师一起编写了第5章,窦延平老师对第5,6,8章进行了修改和补充。在编写完成之后,最后由窦延平老师进行了统稿和审校。虽然如此,由于编者水平有限,错误在所难免,欢迎读者批评指正。

在本书的编写过程中得到了上海市教学名师、计算机系侯文永教授,电子信息和电气工程学院的副院长张焰教授的关心和帮助,在此一并致谢。

编者

2005年3月

目　录

1 绪 论

本章将介绍数据类型和数据结构的基本概念、定义以及常用的表示方法，数据类型和数据结构之间的关系，并且回答什么是数据结构的问题。由于从抽象数据类型的概念出发，研究和描述数据结构是强大的潮流，因此本书选用了面向对象的程序设计语言C ++ 作为描述工具。利用抽象数据类型研究数据结构的优点在于它更便于信息隐蔽、实现的独立性和简捷性、说明的精确性、模块化等等。C ++ 的强大的构造数据结构的能力，使得实现数据抽象、过程抽象更加容易。由于C ++ 具有类的继承、模板、重载等技术手段，使得快捷地生成类属数据结构(Generic Data Structures)成为可能。这对于软件的开发和设计、软件复用等具有重要的现实意义。除此之外，还将介绍评价算法优劣的主要标准，即算法的时间复杂性和空间复杂性的级别。在以后的各章中，我们将用它作为评价各种数据结构好坏的标准。

1.1 数据类型与数据结构

数据类型的本质是用于识别一组数据对象的共同特征，这些特征把该组数据对象辨认为同一种类型。数据类型给出了相应数据对象可能的数据值以及作用在这些数据值上的操作的集合。比如，C ++ 提供了数据类型 int，它包含的数据值是一串连续的整数，但个数是有限的；如…，-2，-1，0，1，2，…。除此之外，数据类型还提供了在这些数据值上的操作的集合。如：我们可以在C ++ 程序中直接使用加、减、乘、除等各项操作，而不必自己重新编写实现的程序。像 int 这样的数据类型通常称之为固有数据类型，它是C ++ 语言固有的，可以直接提供给程序员使用。类似的还有数据类型 char，float，double 等等。

数据类型通常分为两类。第一类是原子数据类型，它的数据值不可以再进行分解。如数据类型 int，就是原子数据类型。它的数据值，无法再进行分解，如数据值 16 和 2 就不可以再进行分解了。但有人会说，16 不是可以分解为 1 和 6，而 2 不是可以分解为若干个点的运动轨迹吗？但是，请注意这样分解以后，它们和这里所讲的数据值已经是完全不同的概念了。

另外一种数据类型是结构数据类型(Structured Data Type)，或者称为数据结构。结构数据类型和原子数据类型不同。它的数据值可以进一步进行分解为组成元素，即数据元素(在数据结构的教科书中通常称之为结点)的集合。这些数据元素之间通常具有某种结构，或者换句话说，它们之间具有某种关系。由于数据结构是一种数据类型，所以它具有数据值以及在这些值上的操作的集合。另外，由于数据值可以分解为数据元素的集合，而数据元素也可以进一步分解为原子数据值或者数据元素(这些数据元素之间也具有某种关系)。因此，也可以说，数据结构是一种数据类型，它的数据值可以进一步分解为数据元素的集合，该数据元素可以是原子数据值，也可以是另外一种数据结构，数据元素之间有一个关系(结构)的集合。

假定，我们定义了一种称之为 sample 的数据类型：

```
typedef  int  sample[3];
```

它的每一个值由排成一行的三个整数构成。由三个整数组成的一个集合是 sample 的一个数

据值。由于这些整数是原子数据值,通过指定哪一个整数是第 0 个、第 1 个、第 2 个数据元素,而确定了一种关系或者结构。因此,类型 sample 是一种数据结构。

必须明白数据结构的每个数据值都有一个相应的结构。因此,即使它们具有相同的构成元素,它们也未必相同。如图 1.1 中所示的数据值 A 和数据值 B 是不相同的。

设有三个类型为 sample 变量 a,b,c 和一个类型为 int 的变量 x。即

$$\text{sample } a,b,c;\text{int } x;$$

变量 a 和 b 的值分别为图 1.1 中的值 A 和值 B。这样,我们可以在数据值上完成一些操作,例如:加法 + 操作和赋值 = 操作。注意:在C ++ 语言中实现时,必须对 + 和 = 进行重载。

$$c = a + b;$$

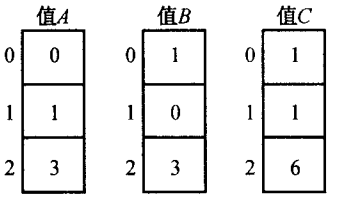


图 1.1 数据类型 sample 的三个数据值

该操作产生一个数据值 C(见图 1.1),并被保存在变量 c 之中。

另一种类型的操作是作用在数据元素上面,而不是作用在数据值上。如操作:

$$x = a[1];$$

其意义为获得变量 a 的数据值中的下标为 1 的数据元素之值,并赋给左边的变量。

这样,我们就知道了数据结构可以有定义在它的数据值之上的操作,也有定义在数据值内的数据元素之上的操作。

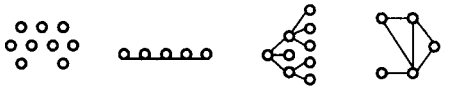


图 1.2 基本结构

由于数据结构几乎都是围绕着结构问题进行组织的,所以结构问题是非常重要的。如图 1.2 所示,通常存在着四种最基本的结构(关系)。第一种是集合。集合中的所有元素或者结点的特征是都属于同一集合。第二种结构是线性结构。它或者为空,或者只有一个结点,否则有一个首结点和一个尾结点。除首结点之外,每一个结点都只有一个直接的前驱结点。除尾结点之外,每一个结点都只有一个直接的后继结点。在该结构中,每个结点关联于另外一个结点是一对一的。第三种结构是树形结构或者层次结构,一个结点可以对应多个结点,或者它们之间的关系是一对多的。最后一种数据结构是图或者网络结构。结点之间的关系是多对多的。集合的结点之间的关系是非常松散的,因此在许多情况下,可以用其他的结构,如线性结构或树形结构来表示它们。如大学的学生记录是一个集合,每一个学生记录都有一个唯一的学号进行标志。这样,我们就可以把这些学生记录按照学号的递增序或递减序,组织成一种线性结构,以有利于查找。为了进一步提高查找、插入、删除一个结点的速度,还可以把它们进一步组织成二叉排序树、平衡二叉排序树等树形结构(请参阅后续章节),从而进一步提高执行基本操作的效率,达到降低时间代价的目的。

1.2 数据类型(数据结构)的实现

数据类型或数据结构的实现分为三步:即抽象数据类型(Abstract Data Type,以下简称 ADT)阶段、虚拟数据类型(Virtual Data Type,以下简称 VDT)阶段以及物理数据类型(Physical Data Type,以下简称 PDT)阶段。在 ADT 阶段,仅仅认为数据类型存在于我们的想象之中,我

们只需要把注意力集中到感兴趣的操作以及数据的取值范畴之中即可,至于实现的细节等统统都可以排除在外。换句话说,此时只需要给出这种数据类型或数据结构的规范即可,即只需给出某一种操作的前提是什么,得到的结论,或者结果是什么,操作的数据对象是什么,它的变化范围是什么,就可以了。比如:对一张名字表的处理,我们可以把该名字表想象为在空中顺序飞行的大雁。它的操作如:查找某一名字、在某个名字之后插入一个新名字以及删除某个特定名字的操作,这些操作的前提条件和操作的结果,都是可以非常容易地构造出来的。事实上,在这一阶段,我们既不关心数据对象在计算机之中的存储形式,也不关心实现这些操作的具体程序代码的细节。因此,在这一阶段,即使是对不懂程序设计的人员来说,理解也是非常容易和便捷的。至于它的实现手段是多种多样的。可以使用自然语言,如中文进行描述;也可以使用数学的手段,如谓词、代数的方法进行描述;同样也可以采用图形的形式来实现;当然,使用程序设计语言,如 C/C ++ ,Java,Ada 等,作为描述手段也是我们经常采用的手段之一。

在 VDT 阶段,我们通常要利用程序设计语言,如C ++ ,写出程序,从而达到保存相应数据和操作的前提和结果的目的。可以利用C ++ 提供的固有数据类型和构造数据结构的手段去构造新的数据类型和数据结构。例如:可以用类名为 strings 的类,表示一张名字表,而允许外部世界对名字表进行的操作,都可以通过公有(public)成员函数进行。这样一来,我们所定义的数据类型和数据结构就好像存在于一个“虚拟的” C ++ 处理器上。该C ++ 处理器,可以看作为C ++ 编译程序同操作系统、计算机硬件的结合体。该虚拟处理器逐条执行C ++ 的语句,并输出相应的数据。我们称这样一种数据类型或数据结构为虚拟数据类型。在用程序设计语言实现 ADT 和 VDT 时,通常并不特意地区分这两个阶段,在有些数据结构的教科书中,都把它们称之为 ADT。为了叙述的简单和方便起见,本书中也采用了这种惯例。

最后,任何一种数据类型和数据结构都必须被存放到物理存储器中,才能通过计算机的基本指令操作得出预想的结果。从数据结构的观点看来,随机存储器(RAM)可以认为是字节的一维数组,无论是 ADT 还是 VDT,最后都必须转化为字节的一维数组才能够执行。我们称在这一阶段的数据类型为物理数据类型阶段。图 1.3 给出了一个简单的总结。

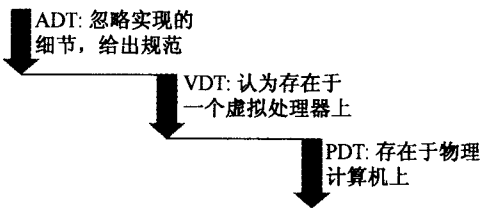


图 1.3 数据类型的三个阶段

从某种意义上来说,ADT 的说明相当于建筑物的蓝图。蓝图是建筑师对于所设计的建筑物的想象的精确描述。施工队必须受蓝图的制约和指导,才能建造出合乎要求的建筑物来。但是,蓝图通常都非常详尽,甚至要告诉施工人员要使用什么材料。但 ADT 的说明或者规范,则与此相反,基本上是一些功能性的说明,对具体的实现不作过多的约束。在具体实现时,所使用的程序的逻辑,所选择的程序设计语言的固有数据类型的多寡以及是否必须设计新的数据类型,完成同样的功能所选取的算法都是自由的。

这样,我们就得到了 ADT 的主要特征:

- 数据类型的内部表示能够被修改,但是并不影响外部程序或函数对程序的访问。数据抽象的方法保持不变。
- 一个 ADT 是一个实体。
- 隐藏在 ADT 之中的一些数据或函数,虽然和外部程序相关,但是这些外部程序并没有

资格去直接访问它们。

图 1.4 给出了 ADT 的表示和实现的过程。组成元素或结点同结构一起构成了它的处理的值域或范畴,然后同其操作构成了 ADT 的规范或者说明。ADT 经过表示和实现之后,给外部程序提供了访问的途径。图中的双箭头给出了外部使用者和内部实现之间的界面。

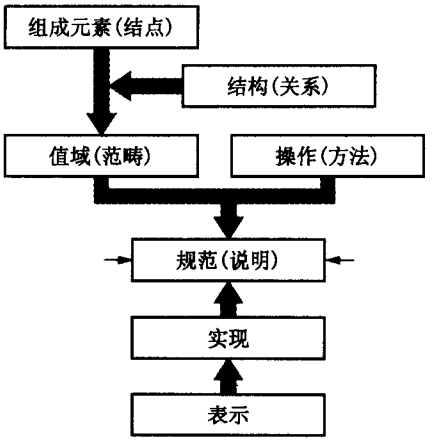


图 1.4 ADT 的形成和使用(其中→←表示用户和实现者之间的界面)

1.3 面向对象的设计和 ADT

面向对象的设计(Object-Oriented Design,即 OOD)是建立在 ADT 的数据抽象的基础之上的。理解 OOD 的最简单的方法是考虑它和传统的面向函数的设计(Function-Oriented Design,即 FOD)的差别。

面向对象的设计是把软件系统作为互相作用的对象的集合,而不是像面向函数的设计中那样作为互相作用的函数的集合。在面向对象的程序设计(Object-Oriented Programming,简称 OOP)中,ADT 之中的数据和操作,都被结合在一起,形成一个单一的实体。这个实体通常称之为对象。面向对象的程序设计语言,如C ++,Smalltalk 和 Object Pascal 都是直接支持对象的使用的,并且简化了 OOD 的实现手段。在C ++ 中,类的构造是用于实现 OOD 的,并且因此定义了对象的类型。

面向对象的程序设计的结构化越强、模块化也越强。它允许程序员更加紧密地接近现实世界的情况。在本节中,首先比较一下传统的面向函数的程序设计方法和面向对象的程序设计方法的特点,然后,将快速地浏览一下C ++ 中的面向对象的程序设计的特点。

1.3.1 面向函数的数据结构

面向函数的数据结构是建立在一系列数据变量和相应的操作之上的,它根本不去关心数据隐藏和保护的问题。面向函数的程序设计方法是建立在以下的模式之上的:

- 定义数据需求;
- 构造数据结构;
- 编写一系列函数去处理这些数据。

传统的程序设计语言如:C,Pascal,Algol 强调的是数据表示和基本数据类型的使用。如:我们看到的用数组表示的数据结构、用记录表示的数据结构。记录表示的数据结构是从基本数据类型派生出来的。而这些技术手段,忽视了用户自定义的数据类型中的数据隐藏问题。

1.3.2 面向对象的数据结构

由于 ADT 被认为是对象的类型,并且 OOD 将现实世界的应用问题分解为相互作用的一系列对象,所以,面向对象的数据结构具有 OOD 的完整的特征。当 OOD 被诸如C ++ 之类的面向对象的程序设计语言实现时,这样的程序就被称之为面向对象的程序。一个面向对象的程序采用的主要手段是类、对象。它们的主要特点如下:

• 抽象数据类型(ADT):ADT 指明了对象的数据和操作,如:它可以通过C ++ 的类的构造来实现。

• 数据抽象:数据抽象提供了一个对象和其他的对象数据之间的接口的手段,而并不包含数据实现时的细节。数据抽象表示一个对象区别于所有其他对象的最基本的特征,给外部访问者提供了一个易于使用的界面。

• 封装和数据隐藏:封装将对象的数据和方法的实现细节隐藏起来。这就避免了那些未经授权的用户访问这些细节的可能性,从而保护了对象的数据信息(包括数据和方法)。这样,封装使数据抽象更加完善。

• 模块化:借助于模块化,逻辑上相关的数据抽象被紧密地结合在一起。

• 类:定义了对象的类型。它保护了数据和操作。

• 对象和类的识别:对象是类的一个实例,它具有一个唯一的变量名。类被用于定义对象的结构和性质。

• 方法:有效的操作,借助于类的成员函数加以实现。成员函数被称之为方法(即操作或动作)。方法被用于发送信息给一个对象。

• 层次和继承:一个大的应用程序可以分解为一系列对象或类。这些类通常具有层次继承关系。基类建立了公共的数据成员和成员函数。子类可以继承基类的成员,并且有能力加以扩充。

• 对象之间的信息传递:对象相互之间是独立的。对象之间可以发送信息。信息可能包含诸如创建一个对象、将一个对象初始化、打印某些信息之类的请求。

• 多态性和动态联编:多态性意味着一个方法可以被用于在派生类中进行不同的操作。在C ++ 中,多态性和动态联编是使用虚函数、重载函数和操作符来实现的。

1.3.3 在C ++ 中的数据结构的构造

在本书中将借助C ++ 提供的类和对象的手段构造数据结构。我们将从一个简单的例子开始,来演示数据结构是如何构造的。更复杂的例子,将在以后的章节中逐步展开。C ++ 提供了一种手段,确保外部程序只能访问允许访问的数据和成员函数,而拒绝访问程序的其余部分。有些信息对用户是可见的,有些则是不可见的。接下来,我们将以一个顺序表为例说明数据结构的构造过程。至于构造数据结构的更高级的一些手段,如:模板、继承、多态性,将随着课程的进展,再作进一步的介绍。

我们将用被称之为 seqlist 的线性表作为例子,它以顺序的方式表示了一些数据元素。我们可以把它理解为一串相邻的货柜,里面装的是信息。参见图 1.5,线性表 seqlist 有 N 项,开始的数据元素的位置序号为 0,而最后一个数据元素的位置序号为 $N-1$。开始的数据元素为表首,最后一个数据元素为表尾。为了简单起见,我们把数据元素仍然称之为结点。设表 seqlist有如下一些基本操作:

Insert:插入一个新的结点到表尾;

Delete:从表中删除具有给定值的结点;

Find:查找具有给定值的结点的位置;

ListSize:给出表 seqlist 所有结点的个数;

DeleteFront: 删除表首结点;

ListEmpty:测试表 seqlist 是否空表;

ClearList:将表 seqlist 所有的结点全部删除,并将表的大小设置为 0。

图 1.5 顺序表 seqlist

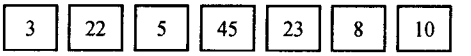


图 1.6 在表 seqlist 尾插入值为 10 的结点

除了上述操作之外,表 seqlist 还允许另外一个操作 GetData,意即知道了结点的位置之后,就可以获得它的数据值。利用这些操作或方法,可以查找表 seqlist 中的结点的最大值。我们可以扫描整个表。从第 0 个位置开始,一直到最后一个结点结束。由于可以通过 ListSize 操作获得表 seqlist 的结点个数,即可知应该搜索多少个结点。参见图 1.7,它说明了如何得到表 seqlist 的结点的最大值。当扫描到表首结点时,最大值取初值 3(即第一个结点的数据值)。当扫描到下一个结点时,它的值 22 比当前的最大值 3 还要大,于是就用 22 取代原来的最大值 3。如此类推,直至得到最终的最大值 45 为止。

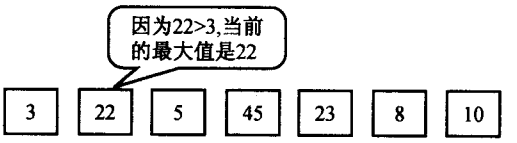


图 1.7 在表 seqlist 中查找结点的最大值的过程

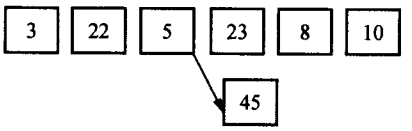


图 1.8 在表 seqlist 中删除值为 45 的结点之后

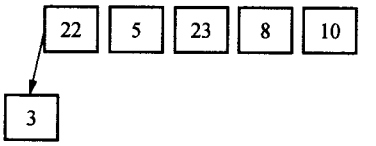


图 1.9 在表 seqlist 中再删除表首结点之后

表 1.1 给出了表 seqlist 的抽象数据类型。在这一阶段,我们只需要总结出表的功能性的说明或规范。它操作的前提、结果是什么,输入是什么,而输出又是什么,这是我们感兴趣的。在此基础上,即可用C ++ 语言描述它的规范,见表 1.2。在表 1.2 中,函数 ListSize,ListEmpty,Find,GetData 都是常成员函数,因为它们并不改变表的状态。而 Insert,Delete,Deletefront,ClearList 则不能作为常成员函数,因为它们改变了表的状态。

表 1.1 表 seqlist 的抽象数据类型(ADT)

ADT seqlist is

Data

一个数据元素的表 seqlist,其数据元素或结点可由序号进行标志。

Operations

Constructor

构造一个空表。

ListSize

前提:无。

结果:给出表 seqlist 的规模。

ListEmpty

（续表）

前提：无。

结果：若表 seqlist 为空返回 True，否则 False。

ClearList

前提：无。

结果：若表 seqlist 非空，则清空表 seqlist。

Find

前提：给定数据元素或结点。

结果：若查找成功则返回 True，否则返回 False。

Insert

前提：给定要插入的数据元素。

结果：将该数据元素插入到表 seqlist 的末尾，表的规模增加 1。

Delete

前提：给定要删除的数据元素之值。

结果：查找该数据值的结点，查找成功则删除之，表 seqlist 的规模减少 1，否则删除失败。

DeleteFront

前提：表 seqlist 非空。

结果：删除队首结点并返回结点的数据值，表 seqlist 的规模减少 1。

GetData

前提：数据元素或结点的位置或序号，其序号在 0 和表的大小减 1 之间。

结果：给出相应表 seqlist 位置序号的结点的数据值。

End ADT seqlist

表 1.2 用 C ++ 表示的表 seqlist 的规范（VDT）

```
//Specification of the seqlist in C ++ ,and Declaration.
class seqlist {
    private:
        int size;  //线性表 seqlist 的大小，当它为 0 时，线性表 seqlist 为空。
        DataType * list;
          //用于保存线性表 seqlist 的所有元素。DataType 为该线性表 seqlist 的数据元素的数据类型。
    public:
        //线性表 seqlist 访问方法。
        seqlist( );
        int ListSize( ) const;
        int ListEmpty( ) const;
        int Find(int & item) const;
        DataType GetData(int pos) const;  //线性表 seqlist 的修改方法。
        void Insert(const int & item);
        void Delete(const int & item);
        void ClearList( );
    };
```