21世纪高等院校教材

数据结构教程

——抽象数据类型描述

（C语言版）

陆松年 编著

科学出版社

21世纪高等院校教材

数据结构教程

——抽象数据类型描述

（C语言版）

陆松年　编著

科学出版社

2002

内 容 简 介

本书是作者在多年教学实践的基础上，为计算机应用专业等非计算机专业的学生编写的一本教材。本书内容深浅适度，语言生动形象，侧重于程序设计技术、算法和应用。

本书从抽象数据类型(ADT)的角度，循序渐进和系统地介绍了线性表、数组、串、栈、队列、树、图和集合等各种基本数据类型的说明、表示和实现，还介绍了查找、排序等各种算法和算法分析方法以及文件的组织结构。

本书不仅可作为大专院校计算机应用专业和电子工程、通信工程、信息处理、管理等非计算机专业学生的教材和教学参考书，还可作为各种成人高等教育学校与计算机有关专业学生的教材、教学参考书和自学读本。本书对于计算机科学专业的师生和从事计算机软件工作的人员也是一本很好的参考书。

图书在版编目(CIP)数据

数据结构教程——抽象数据类型描述(C语言版)/陆松年编著.—北京：科学出版社，2002
(21世纪高等院校教材)
ISBN 7-03-010110-3

Ⅰ.数… Ⅱ.陆… Ⅲ.①数据结构-高等学校-教材②C语言-程序设计-高等学校-教材 Ⅳ.TP311-12

中国版本图书馆CIP数据核字（2002）第007301号

科学出版社 出版
北京东黄城根北街16号
邮政编码：100717
http://www.sciencep.com
源海印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2002年2月第 一 版 开本：B5（720×1000）
2002年2月第一次印刷 印张：25
印数：1—3 000 字数：458 000

定价：32.00元

（如有印装质量问题，我社负责调换〈环伟〉）

前　言

当今的世界已进入了信息时代，计算机的应用也已从传统的数值计算发展到非数值计算，并已逐步深入到各个领域。现在社会各行业大量地用计算机进行数据处理，电子通信、自动控制、信息处理、人工智能、管理和情报检索等各专业与计算机科学的联系日益密切，计算机已不单纯地是一种工具，计算机课程已成了很多非计算机专业课程的重要组成部分之一。

数据结构是计算机学科的一门最重要的基础理论和基础技术课，并已成为很多高等院校非计算机专业的必修课。数据结构课程的任务是介绍使用计算机解题时，根据从各类实际问题归纳、抽象出来的对象的数据特征和对象之间的联系，选择合适的数据组织方法、存储方法和相应的算法。这些方法有助于系统地设计出周密、有效和风格良好的复杂程序。因此，数据结构在系统软件开发和应用软件开发中都具有广泛的应用。

本书参考了国内外最新出版的有关数据结构和算法的教科书和专著，循序渐进和系统地介绍了线性表、数组、串、栈、队列、树、图和集合等基本数据结构，还介绍了文件的组织结构以及查找、排序等各种算法和算法分析方法。本书的各章都配备了大量的习题，最后的数据结构实习一章精选了一些上机实验题。为了使学生能以软件工程的方法分析、设计和编写结构清晰、易读和规范的程序，该章还给出了一个实验范例。

考虑到本书是一本面向计算机应用专业和非计算机专业学生的教科书，因此以简明、实用、通俗易懂为主，侧重于程序设计技术、算法和应用，尽量避免繁琐、深奥的理论推导和证明，只要掌握一门C语言或与C语言相类似的语言就可以读懂本书。

近年来国内外在对计算机专业学生的教学中，以抽象数据类型（ADT）的角度描述数据结构已成了主流。我们认为，一种好的程序设计风格和方法不仅对计算机专业的学生和软件工作者极其重要，对于非计算机专业的学生和应用程序设计员也同等重要。就从程序设计的风格和方法来说，数据结构是不分专业的，只不过是应用的侧重点不同而已。因此，作为主要是为计算机应用专业和非计算机专业而编写的一本教科书，我们也采用了ADT的思想，并将该思想贯穿于全书之中。至今为止，数据结构基本上还是一门有关

程序设计的技术或“艺术”的学科，但 ADT 所涉及的抽象、信息隐蔽和模块化使数据结构这门学科从技术性向科学性前进了一大步，并使两者有机地结合起来。

在所有的算法语言中，C语言是一种数据类型丰富、描述能力强、使用广泛的高级语言，国内外有关数据结构的教材大部分是采用C语言或类C语言。尽管标准C语言不是描述抽象数据类型的理想语言，但基于国内的现实，我们还是用C语言描述和实现各种数据类型的 ADT 说明、表示和实现。为了简明清晰地描述、表示和实现各种抽象数据类型，本书引入了C ++语言的函数参数的“引用”调用方法和类似C ++语言创建对象的 new 操作符(可用C语言的宏定义实现)，并将布尔数据类型作为固有数据类型。读者很容易将这种“类C语言”转化成标准C语言。为了使本书算法描述紧凑、简短，在调用整型、布尔型和 void 型C语言函数的主调程序中，对这些函数的返回值类型一般省略了说明。

本书给出了一个用C++ 语言实现 ADT 的例子，如果读者已经掌握 PASCAL、Ada、Modula-2 和 Java 等面向对象的高级语言，那么就很容易将本书的实现转换成相应语言的 ADT 实现。对于使用不支持记录、指针、动态存储分配和递归过程的 FORTRAN、BASIC 等高级语言的程序设计员，利用本书所介绍的游标、数组和非递归实现的方法，也很容易实现各种数据结构和算法。

本书计划讲课学时为 60。不同专业可根据自己的需要删去或略讲书中某些章节，这样可将学时数压缩为 40～50。值得指出的是，数据结构是一门实践性很强的课程，因此不论在校学生或自学者都应当独立完成本书各章中一定数量的习题，并进行三次以上的上机实习。

全书由陆松年主编，诸葛雷和李晶晖参加了本书第 2 章和第 8 章的编写工作。张殷、刘捷、金斌和赵晶等为本书绘制了大量的图表，并录排了大部分书稿；本书的编写还得到了作者所在学校院、系领导的大力支持，在此深表谢意。

由于作者水平有限，书中错误难免，恳切希望各位学者、专家、讲授该课程的老师和本书读者指出，以便及时更正，作者将不胜感谢。

作　者

于上海交通大学电子信息学院

2001.12.20

目　录

第 1 章　抽象数据类型和算法描述

抽象和抽象数据类型(ADT)在现代的程序设计和编码中起了越来越重要的作用,特别在数据结构的教学中,理解和应用这些概念更显得十分重要。在这一章中,我们将导出抽象和实现的概念,定义抽象数据类型的说明、表示和实现。本章的最后部分还将给出算法的概念和算法描述的方法。

1.1　抽象和实现

抽象的例子在程序设计中出现得很多,尽管你可能还没有认识到这一点。本节中我们先来考虑几个普通的抽象,即数、变量、二维数组和过程。

1.1.1　数和变量

你在程序设计中能够使用整型数和实型数,靠的是抽象,因为事实上在计算机内并不存在这样的数,有的仅是由机器指令操作的二进制位组。计算机懂得你想象成整型的数(在程序中写成以 10 为底的数)与计算机内存中一些 8、16 或 32 个二进制位的联系。

在高级语言程序设计中,经常要使用变量。变量是一种抽象,程序员可以说明一个整型变量,并将值 735 赋给它,如图 1-1(a)所示。在将高级语言程序翻译成目标程序或机器语言程序时,编译、连接和装入程序决定计算机内存中哪一个位置(或地址)对应于该变量,并生成机器指令,在执行目标程序时将 735 这个值赋给 x。假定编译程序分配给 x 的是地址为 1246 和 1247 两个字节的空间,图 1-1(b)描写了经过赋值后的结果。

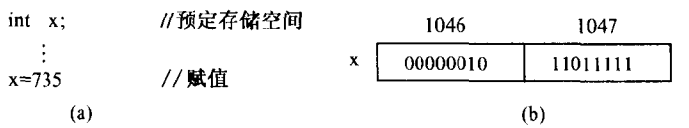

图 1-1　整型赋值的抽象和实现
(a) 程序员所见到的整型赋值抽象　(b) 16 位的空间(二字节)存储了等值于 735 的二进制数

整型量的计算通常由硬件指令执行。为了理解软件一级的抽象，我们来考察实型量的计算。一些计算机在硬件指令集中没有实型或浮点型的指令，或购买者没有购买有关的可选部件。于是如果用某种程序设计语言写了如下赋值语句：x=x*y+3.0，那种语言的编译程序不仅要计算用哪一个存储单元来存储变量 x 和 y，而且还要调用子程序来执行乘法和加法操作。

重要的是通过了使用抽象“变量名”和抽象“实型数”，当写一个赋值语句时，程序员就不必因内存的细节和为了实现所说明的计算而要用到的实际指令而烦恼。抽象是信息隐蔽的方法，用这种方法将数据表示或处理过程中的细节对不需(或不想)了解的人隐藏起来。

本书将经常对照抽象和实现。抽象对于用户即程序设计员是可见的，而实现则是已被隐蔽掉的繁琐的细节。在上面例子中使用了称之为实型数以及包括实型数加、乘和赋值的抽象，还有一个用于说明 x 和 y 是实型的初始化操作。实型数的实现在存储方法上分成尾数和指数两个部分，在操作方面通过机器语言程序调用子程序来实现，这些只要由编译程序设计者来考虑。

在某些语言例如 FORTRAN 中，一些初始操作是隐含的。由于一个变量在首次使用时，编译程序就记住了它，所以不必预先说明。在 FORTRAN 中，实型变量和整型变量的区别可以是隐含的：以字母 I,J,K,L,M 和 N 开头的变量作为整型变量处理，以其他字母开头的变量作为实型变量处理。

1.1.2 两维数组

你可能在程序设计中使用过两维数组，并且也许懂得计算机存储区并不是两维的，而是按字节或字的一维或线性序列方式进行编址。那么很明显，在高级语言程序和计算机线性存储器之间必须有某个中介者，它能正确地解释象 A[3][4]=B[4][7]+1.0 之类的语句，同前例，这也是编译程序。依靠编译程序，你能使用抽象，这样就可获得机器本身所不具备的强有力的表达能力。

用于两维数组的抽象可称之为“矩阵”。矩阵可包括取值(下标变量 B[4][7]处于赋值语句的右边)、赋值(下标变量 A[3][4]处于赋值语句的左边)和初始化(在程序开头对矩阵和它们大小的说明)等操作。也如前述，编译程序的设计者要用线性存储器来实现矩阵，公式中的取值和赋值操作则要由机器语言来实现。

注意，在本例中抽象矩阵已隐含使用了抽象实型数，因为存储在该矩阵中的值是实型数，这说明了抽象本身能嵌套很多层次。

本书将在第 2 章再详细讨论数组，现在先细看一下图 1-2，该图显示了

3 * 4的整型矩阵的抽象和实现。

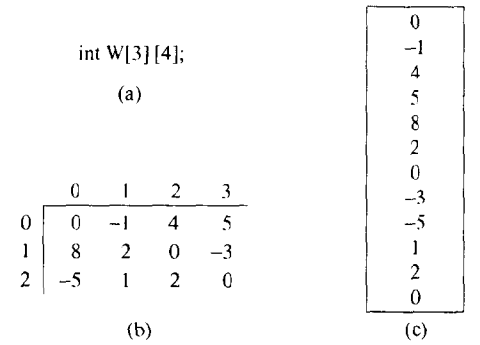

图 1-2 两维数组的抽象和实现

(a) C 说明 (b)程序设计员所看到的已赋值数组 W

(c) 计算机内所存储的已赋值数组 W

1.1.3 过程与抽象

前面所述的抽象都是程序设计语言提供的,抽象的实现则是编译程序的责任。抽象和实现也可以由程序设计员本人来完成。

在大型、复杂的程序设计中,程序员要经常对复杂结构的数据进行特定的运算和处理,这种运算和处理往往是高级语言不能够提供的,如矩阵的加、乘、转置等操作。过程是程序设计的基本工具和方法,它将操作符和操作对象的概念一般化,使程序员不受程序设计语言提供的基本数据类型和操作符的约束,可以利用过程自由地定义操作数和操作方法。

例如,可以在程序中任何需要的地方通过调用过程来进行 n * n 的矩阵乘运算 MatMultiply(A,B,C,n),只要你在适当的地方定义了如下过程:

```
void MatMultiply(MatrixType X, MatrixType Y, MatrixType &Z, int n)
{                          // 矩阵乘操作 Z=X*Y
    int i,j,k;
    for (i=0; i<n; i ++)
        for (j=0; j<n; j ++) {
            Z[i][j]=0;
            for (k=0; k<n; k ++)
                Z[i][j] = Z[i][j]+X[i][k]*Y[k][j];
        }
}
```

如果在你的计算机系统的数学库里有矩阵相乘的过程可供使用，甚至可不必编写该过程，而只要将有关模块连接到代码中。

在大型软件的开发过程中，对一些特定数据类型的复杂操作，可以安排一些人员专门编写和调试所需的模块或过程，调用者只要知道如何使用这些过程，即只要知道与过程的接口方法，而不必知道过程的具体实现细节，这样就将数据的特征和操作与具体的表示和实现的细节分开来，简化了主体程序的设计过程。

利用过程的优点在于信息的隐蔽和模块化，它实现了对算法的若干部分的封装，使其变成了“黑盒子”，调用者看不见也不必看到它的内部结构，而且也无法随意地访问它，只能通过指定的方法，利用所定义的操作才能进行访问。从调用者眼光看来，矩阵的运算与整数和实数的数值运算似乎没有什么特别不同的地方。

由于过程对调用者只指明了其功能，而没有指定要采用何种方法来实现，实现方法是相对独立的。可以用一种技术来实现，也可以用另一种技术来实现，只要满足接口的要求，用户将看不到结果的任何改变。这与用户对两个数相乘的情况是一样的，一台计算机系统可以用移位和加法的重复过程来实现，另一台计算机系统则提供乘法指令直接用硬件实现，而对用户来说，这些差别在他所编写的程序中是看不见的，一般也不必知道这些实现上的细节与差别。由此看来，过程就是一种抽象，而且是更高层次的抽象。

1.2 术语和概念

1.2.1 数据、数据元素和数据对象

1. 数据(Data)

数据是描述客观事物的数、字符以及所有能输入到计算机中并被计算机程序加工的信息的集合。通常意义下的数据，如整数和实数等，只是数据的特例。一个编译程序或文字处理程序的对象是文件中的字符串，所以文件中的字符也是一种数据。对计算机而言，数据的含义是很广泛的，如图形、声音、光和电的输入都是属于数据的范畴。

2. 数据元素(Data Element)

数据元素是数据的基本单位，即数据集合中的一个客体。每一个数据元素可以只有一个数据项(data item)，也可以由若干个数据项组成。数据项是数据的有意义的最小单位。

3. 数据对象(Data Object)

数据对象是性质相同的数据元素的集合，是数据的一个子集。例如整型

数据的对象是集合{0,±1,±2,…},字母字符的数据对象是集合{A,B,…,Z,a,b,…,z},桥牌的数据(点数)对象是集合{2,3,…,10,J,Q,K,A}。数据对象可以是有限的,也可以是无限的。

1.2.2 数据类型

数据类型(或简称类型)是程序设计语言中对于给定变量的所有可能取值的集合。例如,整型变量允许取在计算机硬件中能表示的最大正整数和最小负整数之间的所有整数值。在典型的16位计算机中,整型变量一般取-32 768到+32 767之间的整数值。而一个布尔型的变量仅允许取true和false这两个值。

每一种程序设计语言都有一组固有的或基本的数据类型。在FORTRAN中,基本数据类型是INTEGER、REAL、LOGICAL、CHARACTER和COMPLEX;在PASCAL中,它们是INTEGER、REAL、BOOLEAN、CHAR和指针;在C中它们是int、float、long int、double、char、enum和指针。

很多现代的程序设计语言,例如PASCAL、C、Ada、Modula-2和Java允许定义新的类型。这样的类型可仅仅是对基本类型范围的限制,例如在1与10之间的整型,或基本类型的组合,例如由整型的学生号码、字符串类型的学生姓名和实型的平均学习成绩组成的记录。

每一个对象仅由单值构成的类型称为标量类型或原子类型,如整型、字符型等。程序设计语言提供的基本算术操作,如加、减、乘、除和数学函数大部分是在这种类型上进行的。每一个对象可由一组值构成的类型,例如记录以及数组等,称为组合类型或结构类型(有时也称为数据结构),其数据值可进一步分解为组成元素即数据元素,该数据元素可以是原子类型,也可以是另一结构类型。

1.2.3 抽象数据类型

抽象数据类型(ADT)是一种数据类型及在这个类型上定义的一组合法的操作。抽象数据类型不仅包括数据类型的定义,而且为这个新类型说明了一个有效的操作集合。

一些现代的程序设计语言,如Ada、Modula-2和C++提供了"包"(package)、模块(module)和"类"(class)的结构,允许程序员直接定义ADT,模块中可含有一个或多个抽象数据类型,它可被单独地编译,并能将已建立的ADT作为一个库模块来调用,为外界使用抽象数据类型提供了方便。标准PASCAL的类型定义工具不能定义ADT,但是某些PASCAL的扩充版本如