全国高等院校“十五”计算机规划教材
计算机科学与技术教材系列（8）

数据结构教程

用C++实现的方法

秦小麟　叶延风　高　航　编　著

中国宇航出版社　北京希望电子出版社

全国高等院校“十五”计算机规划教材

计算机科学与技术教材系列（8）

数据结构教程

（用C++实现的方法）

秦小麟　叶延风　高　航 编著

中国宇航出版社
北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

内 容 简 介

这是一部关于数据结构（用C++实现的方法）的实用教科书。内容新颖全面，讲解深入细致，编写时，特别注重根据不同的教学对象定位不同的培养目标，各章、节的重难点，主次内容都做了恰当合理的安排。

全书由10章构成，其主要内容包括：数据结构课程的背景及有关的概念和术语、C++面向对象程序设计要点、线性表、栈和队列、数组、广义表和串、树和二叉树、图、集合和查找、各种常用的排序算法、文件的物理结构及其支持空间数据的索引文件——R树。此外，本书各章均配有一定的算法实例和丰富的习题供读者练习，巩固所学知识。

作者从事一线的教学二十余年，积累了丰富的教学经验，本书在整体结构安排、内容取舍以及整书的编写过程中，都充分考虑了教与学的特点，以及所面对的特定读者的具体需要。在内容上既注重了理论体系的完整性，又兼具系统性和先进性。结构清晰，概念准确，文字叙述简洁明了、可读性强，既便于教师课堂讲授，又便于自学者阅读。通过阅读本书，可对数据结构有全面的了解，并为进一步深入学习和研究计算机科学技术奠定基础。

本书可作为普通高校、高等职业学校计算机科学与技术专业本、专科学生的教材和教学参考书，也可以作为工程技术人员的自学教材或指导书。

出 版 中国宇航出版社
发 行 北京希望电子出版社

社 址 北京市和平里滨河路1号（100013）
北京市海淀区知春路甲63号（100080）

经 销 新华书店
发行部 （010）68372924 （010）68373451（传真）
读 者
服务部 北京市阜成路8号（100830）
（010）68371105 （010）68522384（传真）

承 印 北京广益印刷有限公司
版 次 2003年2月第1版
2003年2月第1次印刷
规 格 787×1092
开 本 1/16
印 张 14.5
字 数 317千字
印 数 1～5 000
本版号 ISBN 7-80144-502-3
本书定价 20.00元

计算机科学与技术教材系列

编 委 会 成 员 名 单

序

近年来计算机科学与技术的发展突飞猛进，其应用范围之广，对国民经济影响之大前所未有，特别是计算机网络、电子商务、多媒体技术等发展，正在彻底改变人类的工作方式与生活方式，同时也彻底改变了传统产业与传统的工作模式。目前，计算机科学与技术是高新技术的主要标志，是先进生产力的重要支柱，因此，发展计算机事业是摆在我们面前的重要任务。有鉴于此，我们特组织编辑了以大学本科学生为对象的《计算机科学与技术教材》丛书，为我国信息化培养人才作一份贡献。

当前，在我国计算机教学及教材建设中普遍存在着一些弊病与不足，主要有如下几种：

1．在计算机教材特别是基础性教材中严重存在知识陈旧、落后，跟不上计算机科学与技术发展的步伐。由于计算机技术的飞速发展，内容更新要求极快，一般三五年就需作重大调整而一二年需作必要调整，而现有教材大都不能适应此种变化速度，这种现象在基础性教材中尤为突出。

2．现有教材很多以国外教材为蓝本，存在着脱离我国具体应用实际的弊病。如何根据我国国情并参考国外先进技术编写教材是当务之急。

3．现有的教材大都适应面窄，多数仅适应计算机本科专业而尤其仅适应少数重点院校与重要院校。而目前，由于计算机教育发展迅速，各种与计算机相近与相关的专业蓬勃发展，在计算机本科专业中也出现了不同层次的要求，特别是以应用型人才为主的培养要求。因此，迫切需要有一套适应面较广的基础性教材以满足多种层次的要求。

根据以上分析，本丛书编写原则是

1．适应计算机科学与技术飞跃发展的需要，本教材丛书具有先进性与时代特性，并且每隔一二年作一次小的调整，每隔四五年重新修订出版。

2．本教材丛书具有适应面广，基础性强的特点，能满足多种层次、多种类型的计算机专业本科学生的需要，特别是满足计算机应用型人才培养的需要。

3．本教材丛书具有密切结合我国应用实际、反映我国计算机事业发展需求的基本特点，并且能在实际应用中发挥作用。

4．教材是有别于一般书籍的一种特殊读物，它要求基本概念清楚，基本理论扎实，知识量大与实际应用联系紧密等特点，它还要求教材的内容逻辑性强，可读性好，深入浅出，并附有习题与参考资料等内容，本教材丛书将突出体现这些特点使其适合于教材需要。

5．本教材丛书选题是根据我国目前实际并参考国际最新动态而制订的。本教材丛书第一批 8 本都是具有普遍性与基础性的教材，在不久我们将分别推出第二批、第三批教材满足不同的需要。

6．本教材丛书聘请有深厚理论基础与应用实践经验且长期在教学、科研第一线工作的高校教授编写，特别是有专长的中、青年教授是本丛书教材编写的主要力量。

我们感谢丛书编委会各位委员为本丛书出版所作出的贡献，我们也感谢北京希望电子出版社为本丛书的立题、编审所作出的努力，最后我们感谢丛书的各位作者为本丛书编写

发行与发展所作出的创造性的和富有成效的工作。

我们还期待广大读者为本丛书提出宝贵意见与建议，我们将通过修订，不断努力把本丛书办得更好。

计算机科学与技术教材编委会

2002 年 3 月于南京大学

前　言

数据结构是计算机专业一门重要的核心基础课，其内容主要是研究和解决非数值计算的问题，讨论如何合理地组织、存储和处理数据的方法与技术。计算机领域内只要涉及到程序的地方，均要用到数据结构的知识；许多课程，例如操作系统、编译原理、数据库和人工智能等也均要用到数据结构的知识。

目前已出现了许多有价值的数据结构教材，早期的描述语言主要是采用抽象语言、PASCAL 语言、类 PASCAL 语言或 PL/1 语言。但是，随着 C 语言在国内外的广泛普及，不仅系统程序员而且越来越多的应用程序员也采用 C 语言作为编程工具。在 20 世纪 80 年代末期，国外出现了用 C 语言作为描述语言的数据结构教材，立刻引起注意。众所周知，C 语言是入门容易，真正掌握其精髓则较难。为跟上教材更新的步伐，保证学以致用，更好地掌握 C 语言以及用 C 语言描述、实现各种数据结构与算法，我们于 1992 年 12 月编写了以 C 语言为描述语言的数据结构教材。从 1993 年起，我校计算机专业的学生就使用以 C 语言为描述语言的数据结构教材，效果很好。

随着面向对象技术的不断成熟，C++已被广泛使用，事实上近年来在应用程序领域使用范围已超过 C 语言。不少学校已把 C++作为第一门程序设计课程的语言，也出现了一些用 C++为描述语言的数据结构教材。虽然我校计算机专业的数据结构教材采用的 C 为描述语言，但在作业，尤其是上机实验和课程设计中，相当比例的学生使用 C++作为编程语言。为保证在新形势下的学以致用，我们着手编写了这本教材，希望能对该课程的教材起到一定的促进作用。

为了保证仅具有 C 语言基础的读者能够顺利使用该教材，我们把 C++的基本内容作为一章介绍。但如果具有了 C++程序设计语言的知识，则可略过此章。

本书详细介绍了各种基本的数据结构：线性表、栈、队列、串、数组、广义表、树和图，以及查找、排序和文件结构等。全书共分为 10 章。第 1 章介绍了数据结构课程的背景及有关的概念和术语,并讨论了抽象数据类型、复合数据类型和递归函数等内容。第 2 章简要介绍了 C++的基本知识。第 3 章至第 7 章分别介绍了上述的各种数据结构。第 8 章介绍了各种基本的查找方法，同时，还介绍了一种对算法进行简单时间复杂性分析的方法。第 9 章讨论了各种常用的排序算法。第 10 章讨论了文件的物理结构，并介绍了支持空间数据的索引文件——R 树。在各章中均配有一定的算法实例，借以训练灵活使用所学数据结构的能力。

本书的主要特色是没有简单地把一些算法从其他描述语言移植到 C++，而是根据 C++的特点设计与实现了各种数据结构的存储结构和算法。因此，本教材有助于读者进一步掌握 C++的编程思想、方法和技术。

本书可作为计算机专业的本科生教材，学时安排大约为 70～80 学时。数据结构是一门实践性很强的课程，故每章均附有一定量的习题，只有通过大量的编程及上机实习才能更好地熟练掌握该课程的内容。除此之外，还可以通过该课程的“课程设计”（大约一周时间），把本教材内容尽可能多地溶进一个大的上机实习题中，以加深、巩固在课堂上所学的内容。根据我们的教学经验，在该课程的教学中必须要求学生养成良好的编程风格，尽量多加注

释语句，最好在具体编程前先给出其算法思想。学习本书前，读者应具有C语言或C++的基础，若有离散数学和概率论的知识则更好。

本书第1，8，10章由秦小麟编写，第2，4，6，9章由叶延风编写，第3，5，7章由高航编写。全书由秦小麟统稿，李立新教授审阅了本教材的初稿，并提出了许多宝贵的意见。在编写该教材过程中，赵宝献做了大量的排版与输入工作，在此表示衷心的感谢。

由于编者水平有限，书中难免有错误及不妥之处，恳请读者与同行批评指正。

编　者

2002年8月于南京航空航天大学

目　录

第1章
绪　论

1.1　数据结构课程内容及其意义

数据结构不仅是计算机专业教学计划中的核心课程，而且被越来越多的与计算机应用有关的专业作为重要的选修课或必修课。在20世纪60年代，数据结构没有作为独立的一门课程来开设，它的大部分内容散布在表处理语言、图论、离散数学结构等教材中。1968年美国Donald E.Knuth出版了专著《计算机程序设计技巧》第1卷《基本算法》，首次系统地阐述数据结构的主要内容，即数据的逻辑结构、存储结构以及对数据进行各种操作的算法。到20世纪70年代中期和20世纪80年代初各种数据结构著作大量问世，我国于20世纪70年代末开设了数据结构课程。

用计算机解题，首先要把客观对象抽象为某种形式的数据，然后设计对这些数据进行操作的算法，由计算机执行这些算法，最后获得问题的解答。例如，用计算机来处理高等学校招生录取的工作。首先把考生抽象为姓名、各科考试成绩、总分等数据项组成的记录（一种数据结构）；然后设计对考生记录进行操作的算法，如按总分进行从高到低排序。这里我们可以看到解题过程中要解决两个问题：数据结构的设计和算法的设计。Niklaus Wirth的著作取名为“算法＋数据结构＝程序”点破了计算机解题的真谛。他认为程序就是在数据的某些特定的表示方式和结构的基础上对抽象算法的具体描述。数据结构和算法是程序设计过程相辅相成、不可分割的两个方面。不了解施加于数据上的算法就无法决定数据结构，反之算法的结构和选取往往很大程度上依赖于数据结构。

由于计算机应用的初期侧重于解决数值计算的问题，处理的对象相对简单，如处理整数运算、实数运算等，用简单变量或数组这些形式的数据结构足以表示要解决的问题。因此，那时的程序设计重点是制定最佳的算法。但是随着计算机应用扩展到非数值计算领域，如行政事务处理、人工智能模拟、实验数据的采集与处理等，处理的数据量和复杂度大大增加了。合理地组织、存储和处理数据的重要性已越来越明显地显露出来。现在有关数据结构的知识已成为设计和实现编译系统、操作系统、数据库管理系统及其他系统程序和一些应用系统的重要基础。

1.2　基本概念及术语

1. 数据（Data）与信息（Information）

数据是指描述客观事物的数、字符、正文、图形、图像和语音等所有能输入到计算机中被计算机处理的符号集合。这些符号集合最终在计算机内部表示为二进制位串的形式。所以数据是待计算机加工的二进制位串，它是计算机加工的“原料”。信息则是经计算机加工以后产生的有意义的数据，它是加工后产生的“产品”。习惯上，数据和信息两词互相通用，但实际上是有区别的。例如经过摄像机摄取的图像信号输入到计算机中成了未加工的数据，经过计算机加工（去噪声，增强等）形成清晰的图像，它能给予人们有意义的信息。

2. 数据项和数据元素

数据元素是数据的基本单位，相当于物质的“分子”。比如学生成绩登记表是一种数据结构，它由许多学生成绩记录组成。成绩记录则是构成登记表的元素，即“分子”。有时数据元素又由许多数据项组成。数据项是数据的最小单位，相当于“原子”。例如，每个学生成绩记录中可能含有姓名、课程名、成绩等数据项。

3. 数据对象

数据对象是具有相同特性的数据元素的集合，例如字母的数据对象为

$$C=\{A, B, C, \ldots, Z\}$$

4. 数据结构

数据结构是带有某种结构的数据元素的集合，即数据元素之间存在某种形式的联系。例如，任意两个数据之间可以区分为哪个在前哪个在后，即数据元素之间存在“有序”的结构。在不同类型的数据结构中，它们的数据元素之间可有不同形式的联系，即存在不同的结构。读者可以在今后各章中进一步了解其内涵。

5. 物理结构

物理结构是数据结构在计算机中的影射。在实现中必须考虑两种影射，一是数据元素的映射，即存储具体的每一个数据元素；二是数据间关系的映射，即表示数据元素间关系。根据具体的应用，可以选择不同的存储结构来反映数据元素间的关系，例如顺序结构、链表结构、树结构、图结构等。

1.3 抽象数据类型及面向对象概念

抽象数据类型（Abstract Data Type，ADT）是带有一些操作的元素的集合，它是一种描述用户和数据之间接口的抽象模型，抽象数据类型提供了一种用户定义数据类型的手段。我们知道，任何一个数据类型必须有其一组相应的操作，否则该类型在具体的应用中是没有意义的。作为一个抽象数据类型，其构成的两要素为数据的结构和相应的操作集合。

实际上，在高级语言中已经反映了该思想。例如在 C++中，带有加、减、乘、除运算的整数集合就是一抽象数据类型，具体表示为 int 类型，它是整数抽象数据类型的软件模型。抽象数据类型的软件模型最重要的是其函数性质，而不必关心其实现细节。它们的内部结构与实现有关，但与用户无关。抽象数据类型的优点主要是，如果改变了它们的基本数据结构而不需要改变使用抽象数据类型的程序，抽象数据类型改善了可移植性。

数据结构是表示抽象数据类型的特定方法，通常对一个问题可以有几种方法。例如有理数可以表示为浮点变量（float），或者表示为由一对整数（int）组成的一个结构。

数据类型有以下三类。

- **原子数据类型**

这些数据类型存放单个数据值。通常 C++内部定义的数据类型适合于这种类型的数据。但是有时也需要建立自己的原子数据类型，例如人们需要有时包含几十位数字的整数，它已超出了 C++语言内部整数型的表示能力。

■ **固定聚集数据类型**

这种类型的值含有固定个数的原子分量。例如“复数”抽象数据类型可以看成是一对实数分量，一个是实部，一个是虚部。这种数据类型通常用结构或数组表示。

■ **变聚集数据类型**

这种数据类型的值含有可变个数的分量。例如抽象数据类型“有序整数表”含有任意长的有序整数序列。虽然数组有时能表示这种抽象数据类型，但通常比较适合用用户定义的数据结构来表示，它可以动态地分配存储空间并用指针将它们链接在一起。本书主要处理这种数据类型。

抽象数据类型的一个重要成分是一组与该类型有关的操作，这些操作用以建立、操纵、删除或处理特定类型的数据对象。在C++中，实际上抽象数据类型可以分为两类：内部抽象数据类型和用户定义抽象数据类型。例如，整数是用类型int说明的变量而建立的，是用操作符+，-，×和/等进行操作的。有关int类型的数据结构和相应的操作集合，均由C++编译器提供，所以在使用int变量时不必关心它们是如何存储或者操作符是如何工作的。但是用户定义的抽象数据类型的变量，则必须由用户定义其数据结构和相应的操作（或函数），只有通过这些操作（或函数）对变量中的数据进行处理。

在C++中，往往使用用户定义的类（Class）来表示抽象数据类型。类由存储数据的成员集合和方法（对数据进行的操作）集合构成，用类说明的变量称之为对象。可以看出：类的成员对应了抽象数据类型的数据结构，类的方法对应了抽象数据类型的操作。C++的类分为两部分：私有（Private）部分和公共（Public）部分，前者定义了类的数据结构和内部操作，后者定义了用户使用该类的操作集合。私有部分的成员只能通过类内部的方法进行操作，所以数据通常定义在私有部分，以防止外部非正常对数据的操作。

抽象数据类型或类的优点是类将数据与方法封装（Encapsulation）在一起，其内部实现细节对外部是隐藏的，即用户不必了解使用的类的数据是如何存储的，操作是如何实现的，只需通过类的公共部分提供的操作即可使用该类。从类本身来说，在结构上隐藏了实现细节并控制了外部对数据的操作，有利于数据的完整性，同时也有利于程序的复用。

例1-1 用抽象数据类型来实现“复数”的表示与操作。

有多种方法表示复数，例如可以用笛卡尔坐标：$z=\alpha+i\beta$，这里α，β是实数，α是实部，β是虚部。也可以用极坐标：$z=re^{i\theta}$，这里$r=\sqrt{\alpha^2+\beta^2}$，$\theta=\tan^{-1}(\beta/\alpha)$。这两种方式很容易进行转换,选哪一种方法可依各人所好。在本例中采用笛卡尔坐标。

实现这个抽象数据类型的函数有：存储一个复数，取复数的实部和虚部，复数加法和乘法。关于复数减法、除法和测试相等的函数留作习题。

下面给出复数抽象数据类型表示的类定义：

```
template < class Type> class Complex
{
public :
   LoadComplex (Type *p_complex, double real, double imaginary);
   //存储一个复数
   GetComplex(Type *p_complex, double *p_real, double *p_imaginary);
   //取复数的实部和虚部
```

```
    AddComplex(Type *p_sum, Type *p_complex1, Type *p_complex2);
    //复数相加，p_complex1,p_complex2 存放两个相加的复数，p_sum 存放两者之和
    MultiplyComplex(Type *p_product, Type *p_complex1, Type *p_complex2);
    //复数相乘，p_complex1,p_complex2 存放两个相乘的复数，p_product 存放之积
private:
  double real;   //实数的实部
  double imaginary; //实数的虚部
}

template <class Type> Complex <Type>::LoadComplex(Type *p_complex, double real,
double imaginary)
{
//存储一个复数
      p_complex→real = real;   //存储复数的实部
      p_complex→imaginary = imaginary;   //存储复数的虚部
}

template <class Type> Complex <Type>::GetComplex(Type *p_complex,
double *p_real, double *p_imaginary)
{
//取复数的实部和虚部
      *p_real = p_complex→real;                  //取复数的实部
      *p_imaginary = p_complex→imaginary;   //取复数的虚部
}

template <class Type> Complex <Type>::AddComplex( Type *p_sum, Type *p_complex1,
Type *p_complex2)
{
//复数相加，p_complex1,p_complex2 存放两个相加的复数，p_sum 存放两者之和
//p_sum = p_complex1 + p_complex2
//        = (a + bi) + (c + di)
//        = (a + c) + (b + d)i
    p_sum→real = p_complex1→real + p_complex2 →real;   //计算和的实部
    p_sum→imaginary=p_complex1→imaginary+p_complex2→imaginary;//计算和的虚部
}

template <class Type> Complex <Type>:: MultiplyComplex(Type *p_product,
Type *p_complex1, Type *p_complex2)
{
//复数相乘，p_complex1,p_complex2 存放两个相乘的复数，p_product 存放之积
// p_product = p_complex1 * p_complex2;
//           =(a + bi) * (c + di)
//           =(ac–bd) + (ad + bc)i
```