21世纪 高等学校电子信息类专业规划教材

数据结构(C++语言描述)

丁芝芳 刘 杰 主 编
谭 锋 副主编

清华大学出版社
http://www.tup.tsinghua.edu.cn
北京交通大学出版社
http://press.bjtu.edu.cn

21 世纪高等学校电子信息类专业规划教材

数据结构(C++语言描述)

丁芝芳　刘　杰　主编

谭　锋　副主编

清华大学出版社

北京交通大学出版社

• 北京 •

内 容 简 介

本书是一部关于数据结构(C++语言描述)的全新的教材。内容新颖全面，讲解通俗易懂，结构清晰合理。编写时通过贴近实际的事例和清晰的图示表现数据结构课程的内容及相关的算法思想，以求激发学生掌握专业基础理论的兴趣和满足学生实际应用的需要。

全书共 8 章，包括绪论、线性表、栈和队列、数组和广义表、树和二叉树、图、查找、排序等内容。各章根据不同的教学目标，恰当合理地安排了内容层次及应用实例和相应的习题。

本书总结了作者一线教学 20 余年的经验，注重研究教与学的特点，充分考虑学生的需求。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、高等职业技术学校计算机类各专业、信息类及相关专业本、专科学生的教材或教学参考书，也可供非计算机专业学生选用，同时希望对自学计算机软件开发的人员也有所帮助。

图书在版编目（CIP）数据

数据结构：C++语言描述/ 丁芝芳，刘杰主编；谭锋副主编. —北京：清华大学出版社；北京交通大学出版社，2004.7

（21 世纪高等学校电子信息类专业规划教材）

ISBN 7-81082-250-0

Ⅰ.数… Ⅱ. ①丁… ②刘… ③谭… 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. ①TP311.12 ②TP312

中国版本图书馆 CIP 数据核字（2004）第 051304 号

责任编辑：陈川

出 版 者：清 华 大 学 出 版 社　　邮编：100084　　电话：010-62776969

　　北京交通大学出版社　　邮编：100044　　电话：010-51686045，62237564

印 刷 者：北京鑫海金澳胶印有限公司

发 行 者：新华书店总店北京发行所

开　　本：185×260　　印张：18　　字数：413 千字

版　　次：2004 年 7 月第 1 版　　2004 年 7 月第 1 次印刷

书　　号：ISBN 7-81082-250-0/TP・125

印　　数：0 001～5 000 册　　定价：24.00 元

前　言

数据结构是计算机专业和信息管理专业一门重要的专业基础课，其主要内容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；其主要任务是使读者掌握数据组织、存储和处理的常用方法和常用的算法思想及在实际中的应用技巧，为今后学习后续专业课和进行软件开发打下良好的基础。只要涉及程序的地方，均要用到数据结构的知识，许多课程（如操作系统、编译原理、数据库和人工智能等）也要用到数据结构的知识。

已出版的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C语言等。随着面向对象技术的不断成熟，C++已被广泛应用，现在越来越多的学校已把C++作为第一门程序设计课程的语言。由于C++语言在应用程序设计中的广泛使用，用它来讲授数据结构和算法课程是比较方便的。

本书详细介绍了各种基本的数据结构，包括线性结构、数组和广义表、树（层次）结构和图结构；基本的存储结构，包括顺序（数组）结构、链接结构、索引结构和散列结构，以及其中的一种或多种结构的组合及应用。介绍了各种结构的基本操作，包括插入、删除、查找和排序等操作；还介绍了算法时间复杂度的分析方法、算法空间复杂度的分析方法、递归方法及分而治之方法的应用。

本书力求改变读者对数据结构课程内容抽象的惧怕和认为数据结构课程无用的错误观点，以通俗的语言和实例讲授了数据结构课程的内容。实例丰富，图表清晰，深入浅出，易于学生融会贯通。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，有助于读者进一步掌握C++的编程思想、方法和技术内涵。

本书是作者多年教学经验的结晶，与以往教材比较有创新之处。特别是本书得到了北京工商大学刘杰副教授的指点，对各章节提出了大量的修改意见，丰富了教材内容。本书希望能够引起读者对数据结构课程的兴趣，提高对数据结构课程的重要性、必要性的认识，特别希望能够提高数据结构课程的教学实效，让读者满意并有收获。

由于编者水平有限，书中错误和不妥之处在所难免，恳请读者与同行批评指正。

丁芝芳

2004年6月

目　录

第1章 绪 论

也许你已经从程序设计语言课程中学到了“程序”这个术语，也许你已经熟悉很多应用程序并能熟练地使用，也许你也进行过程序设计，也许……毫不否认，“程序”这个术语大家是耳闻目睹，可是你真的知道“程序＝算法＋数据结构”的含义吗？

为了编写出一个好的程序，必须分析待处理对象的特性及各处理对象之间的关系。这就是数据结构学科形成和发展的背景。

1.1 程序＝算法＋数据结构

1.1.1 程序的产生及对程序的分析

1. 程序产生的五个阶段

需求(requirements)：充分了解所提供的信息(输入)及将要产生的结果(输出)，对所有输入及输出进行严密的描述。

设计(design)：根据需求所得的数据集(如迷宫、多项式等)，编写解决问题的算法；算法不必拘泥于形式，可以用所熟悉的伪语言、图、文字或程序设计语言来表示。

分析(analysis)：针对问题提出其他解决方案，最后，在所有可能的算法中挑选最佳者。

细化(refinement and coding)：细化、修改所选择的算法，配合使用的程序设计语言特性编写出程序的初稿。

验证(verification)：验证工作包括程序验证(program proving)、测试(testing)及调试(debugging)三部分。

简单的程序如下所示。

```
int main()
  {
  string  str;
  cout<<"Hello, My students!\n";
  cout<<"Please enter your name\n";
  cin>>str;
  cout<<"Hello, "<<str<<"!\n";
  }
```

2. 对程序的分析

程序是由一些指令组合而成，而如何将指令进行适当地组合使其解决问题是编写程序的目的。对程序的整体性能进行分析通常要考虑两个因素，即执行时所花费的时间(time)和执行时所需要的内存空间(space)。而时间是比较重要的因素，因为通常无法准确地计算执行一个算法所需要的时间。一般来说，一个程序所需要的执行时间受到下列因素的影响。

- 程序所输入的数据量的多少。
- 程序所使用的算法，算法的时间复杂度(time complexity)。
- 编译器所产生的机器代码(machine code)是否最优。
- 指令在计算机中的执行速度。

其中，后两个因素与所使用的计算机硬件有关。

程序除了要能被执行外，一个好程序的目标如下。

- 执行结果正确。
- 执行效率高。
- 可读性强(程序说明详细，变量名称恰当)。
- 易于修改。
- 易于扩充。

1.1.2 算法的概念及对算法的要求

当程序设计人员对将要解决的问题描述得比较清楚以后，如果想要利用计算机来解决问题，那么就得编写程序，可是，有些复杂的问题直接开始写程序会有些困难，可以先从算法(algorithm)入手，再配合适当的数据结构，这样就能很容易地写出程序。这里所谓的算法是指在有限的时间范围内，解决某一问题的一系列指令的集合。算法必须满足以下几个条件。

- 输入：具有 0 个或多个输入的外界量(算法开始前的初始量)。
- 输出：至少产生一个输出，它们是算法执行后的结果。
- 有穷性：每条指令的执行次数必须是有限的。
- 确定性：每条指令的含义都必须明确，无二义性。
- 可行性：每条指令的执行时间都是有限的。

因为对某个特定问题通常存在多种解法，所以可以用几种不同的算法去解决同一问题，到底选取哪一个算法比较好取决于很多因素。一个好的算法通常考虑以下几点作为目标。

1. 正确性

算法应当满足具体问题的需求，因此正确性是设计和评估一个算法的首要条件，如果一个算法不正确，即不能完成或不能较好地完成所要求的任务，其他方面也就无从谈起。一个正确的算法是指在合理的数据输入下，能够在有限的运行时间内得出正确的结果。

2. 可读性

算法主要是供人们阅读与交流，其次才是机器执行，因此可读性是指一个算法供人们阅读时容易理解的程度。一个可读性好的算法，应该符合结构化和模块化程序设计的思想；

应该对其中的每个功能模块、重要数据类型或语句加以注释；应该建有相应的文档，对整个算法的功能、结构、使用及有关事项进行说明。

3. 健壮性

当输入数据非法时，算法也能适当地做出反应或进行处理，而不会产生奇怪的输出结果。因此，健壮性是指一个算法对不合理(又称不正确、非法、错误等)数据输入的反应和处理能力。一个好的算法应该能够识别错误数据并进行相应的处理。对错误数据的处理一般包括打印出错误信息、调用错误处理程序、返回标识错误的特定信息、中止程序运行等。

4. 时间效率与低内存需求

时间效率指的是算法执行时间；低内存指空间占用少，并且合理。

1.1.3 算法与程序的关系及算法的描述方法

1. 算法与程序的关系

算法的含义与程序十分相似，但二者又有区别。程序中的指令必须是机器可以执行的，而算法中的指令则无此限制；程序有变量说明、初始化、输入、输出，算法可以省略或简化这些烦琐的过程。

2. 算法的描述方法

一般来说，算法可以使用各种不同的方法来描述。最简单的方法是使用自然语言，用自然语言来描述算法，其优点是简单且便于人们对算法的阅读，缺点是不够严谨。还可以使用程序流程图及N-S图等算法描述工具，其特点是描述过程简洁、清晰明了。用这些方法描述的算法不能直接在计算机上执行，若要将它转换成可执行的程序还有一个编程的问题。

可以直接使用某种程序设计语言来描述算法。在数据结构课程中，算法均用函数来描述。在本书中采用C++语句描述算法，它的优点是类型丰富且语句精练，具有面向过程和面向对象的双重特点，编出的程序结构化程度高，可读性强。

【例1.1】 要求从左至右检查数组 $a[0:n-1]$ 中的元素，以查找与 x 相等的那些元素。如果找到一个元素与 x 相等，则返回 x 第一次出现所在的位置。如果在数组中没有找到这样的元素，则返回-1。其实现的算法如下。

```
template<class T> int SequentialSearch(T a[], const T & x, int n)
{                    // 在未排序的数组 a [ 0 : n-1 ]中搜索 x
                     // 如果找到，则返回所在位置，否则返回-1
int i;
for (i=0; i < n && a[i] !=x; i++);
if (i==n) return -1;
return i;
}
```

1.1.4　数据结构简介

首先分析数据的特点，然后选择适当的数据结构，再配合算法(algorithm)，这样就能很容易地写出程序。用计算机解题可以按照这样的思路，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，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。首先把考生抽象为准考证号码、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等。

这里可以看到实现过程要解决两个问题：数据结构的设计和算法的设计。Niklaus Wirth将其归纳为“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中相辅相成、不可分割的两个方面。

那么什么是数据结构呢？数据结构用来研究计算机内部的各种数据存储方式，研究如何有效地维护、处理和应用数据，如何分析整理源数据，建立数据间的相互关系，以最有利的形态存放在内存里以便计算机处理，并提供一种策略使计算机能充分地操纵这些数据。概括地说，数据结构是组织和访问数据的系统方法。一般来说，对数据结构有如下要求。

- 保持数据项与数据项的先后次序及相互关系。
- 使数据所需的存储空间最小。
- 使数据的存取时间最短。
- 提供最有利于用户的环境、最好的界面。

其他还有必要的技巧、算法和策略。要理解数据结构本身，重要的是理解实现数据结构操作的算法。

1.2　数据结构的基本内涵

1.2.1　数据结构研究的内容

数据结构包含三个方面的内容，即数据的逻辑结构、数据的存储结构和对数据所进行的操作。

- 数据的逻辑结构——数据之间的相互关系。
- 数据的存储结构——数据的逻辑结构在计算机中的表示。
- 操作算法——插入、删除、修改、查询、排序等。

这三个方面的关系如下。

- 数据的逻辑结构独立于计算机，是数据本身所固有的，它与数据存储无关。
- 数据的存储结构是逻辑结构在计算机存储器中的映像，必须依赖于计算机。
- 操作是指对数据所进行的一组操作的总称。操作的定义直接依赖于逻辑结构，但

操作的实现必须依赖于存储结构。

1.2.2　基本概念和术语

1. 数据

数据是对客观事物的符号表示，在计算机科学中数据是指所有能输入到计算机中并能够被计算机程序所处理的符号的总称。

2. 数据元素

数据元素是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

3. 数据项

数据项是数据不可分割的最小单位。

4. 数据对象

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，整数数据对象是集合 $N=\{0, \pm1, \pm2, \cdots\}$，字母字符数据对象是集合 $C=\{'A', 'B', \cdots, 'Z'\}$。

5. 逻辑结构

数据逻辑结构是数据相互之间存在一种或多种特定关系的数据元素的集合。数据的逻辑结构简称数据结构。

根据数据元素之间关系的不同特性，通常有如下 4 类基本的数据结构(见图 1-1)。

- 集合结构：结构中的数据元素之间除了同属于一个集合的关系外，别无其他关系。
- 线性结构：结构中的数据元素之间存在一对一的关系。
- 树型结构：结构中的数据元素之间存在一对多的关系。
- 图状结构(网状结构)：结构中的数据元素之间存在多对多的关系。

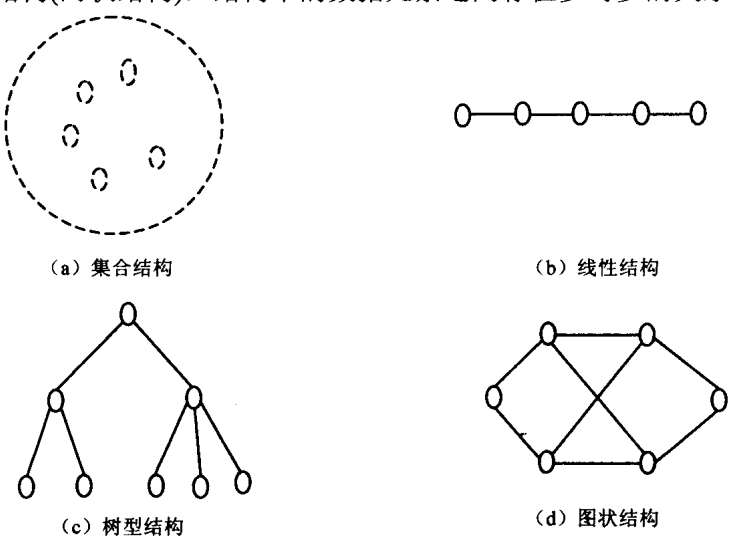

图 1-1　基本数据结构示意图

6. 数据类型

数据类型是与数据结构密切相关的一个概念。它是一个值的集合和定义在这个集合上

的一组操作的总称。

7. 抽象数据类型

抽象数据类型是指一个数学模型及定义在该模型上的一组操作。抽象数据类型的定义仅取决于它的一组逻辑特性，而与其在计算机内部如何表示和实现无关，即不论其内部结构如何变化，只要它的数学特性不变，都不影响其外部的使用。

抽象数据类型和数据类型实质上是一个概念。例如，各个计算机都拥有的“整数”类型是一个抽象数据类型，尽管它们在不同处理器上实现的方法可以不同，但由于其定义的数学特性相同，在用户看来都是相同的。

1.2.3 数据的存储结构

数据的存储结构是指某一种数据结构在存储器中的存储方式，也称为数据的物理结构。数据的存储结构可分为：顺序存储(向量存储)、链式存储、索引存储和散列存储。

1. 顺序存储

一般每一个存储结点只含一个数据元素，所有存储结点相继存储在一个连续的存储区里，一般用存储结构间位置关系表示数据元素之间的逻辑关系。

2. 链式存储

每一个存储结构不仅包含一个数据元素，还包括指针。每一个指针指向一个与本结点有逻辑关系的结点，即用指针表示逻辑关系。

3. 索引存储

每一个存储结点仅含一个数据元素，所有的存储结点都连续存放。此外，增设一个索引表。

4. 散列存储

每一个存储结点仅含一个数据元素，数据元素按散列(Hash)函数确定存储位置。

1.2.4 数据结构的表示

1. 数据结构的二元组表示

为了更确定地描述一种数据结构，通常采用二元组表示，即

$$B=(K, R)$$

其中，B 是一种数据结构，它由数据元素的集合 K 和 K 上的二元关系 R 所组成。有

$$K=\{k_i \mid 1\leqslant i\leqslant n,\ n>0\}$$

$$R=\{r_j \mid 1\leqslant j\leqslant m,\ m>0\}$$

其中 k_i 表示集合 K 中的第 i 个数据元素，n 为 K 中数据元素的个数。

注意　若 $n=0$，则 K 是一个空集，因而 B 也就无结构而言，有时也可以认为它具有任一结构。

---

2. 数据结构的抽象形式

为了理解数据结构的设计，可以使用抽象数据类型(ADT)模型来指定数据存储类型及

支持数据的操作。

抽象数据类型可以用三元组表示(*D*, *S*, *P*)，其中，*D* 是数据对象，*S* 是 *D* 上的关系集，*P* 是对 *D* 的基本操作集。

抽象数据类型的定义格式如下。

```
ADT 抽象数据类型名{
    数据对象：<数据对象的定义>
    数据关系：<数据关系的定义>
    基本操作：<基本操作的定义>
}ADT 抽象数据类型名
```

抽象意味着应该从与实现方法无关的角度研究数据结构，通常人们只关心数据结构能做什么，而不关心它是如何实现的。在程序中使用数据结构之前，必须提供实现方法，应该更关心操作时的运行效率。以 ADT 的形式表示数据结构，程序设计人员便可将精力集中在数据及其操作的理想模型上。ADT 操作描述如下。

操作名称：指定输入参数、数据结构元素的操作类型和输出值的动作语句。

前提条件：输入参数必须满足的条件及操作成功执行的对象状态。

后置条件：操作引起的数据结构中数据的变化。

ADT 的主要功能是数据结构属性和操作的封装。操作是指接收数据值(参数)、执行计算并返回值的过程。数据值是操作的输入，而返回值是操作的输出。为了指明操作执行的任务，使用动作语句来说明函数如何接收输入参数、执行指定的计算以及如何返回输出值。在某些情况下，可能只有当满足某些条件时，数据结构才成功地执行操作。操作描述中还包括前提条件和后置条件。如果操作没有前提条件或后置条件，ADT 操作描述将忽略这些条件。要理解数据结构，重要的是理解实现数据结构操作的算法。

【例 1.2】 以 ADT 形式表示的圆是确定圆形半径、面积和圆周的计算工具。它的对象以实数存储半径并具有以下的操作。

getRadius(): 返回圆的半径。

setRadius: 设定半径为输入参数 *r* 的数值。前提条件：新的半径 *r* 必须为正数。后置条件：新的半径具有值 *r*。

area: 使用公式 $\pi \times r^2$ 计算并返回圆的面积。

circumference: 使用公式 $2 \times \pi \times r$ 计算并返回圆的周长。

写出类的声明。

将构造函数作为外部函数实现，使用构造函数初始化表。

将函数 setRadius()作为外部函数实现。

使用嵌入码实现 circle 类。使用下面的 PI 常量声明：

```
const double PI=3.141 592 653 589 793
```

解题过程的程序代码如下。

(1)