计算机与信息技术专业应用教材

数据结构教程

（C语言版）

李春葆 金晶 编著

清华大学出版社

▶计算机与信息技术专业应用教材

数据结构教程（C语言版）

李春葆　金晶　编著

清华大学出版社

北京

内 容 简 介

本书根据高等院校计算机专业数据结构课程的教学大纲要求，结合十年战斗在一线教授的丰富教学经验编写而成。全书按照课程的授课顺序，阐述了线性表、栈和队列、串和数组、树和二叉树、图、查找、排序等内容。

本书注重实用性和可读性，对概念原理的阐述准确、精练、通俗易懂；在介绍数据结构的基本运算时，不仅介绍了算法思想，更注意程序的实现过程；源程序都经过上机验证，正确率高；每章最后都配备了大量的习题，并在附录中给出了详细的习题答案，使学生能够深化对基本概念的理解，提高分析与解决问题的能力。

本书适合于作为计算机及相关专业应用型本科或专科的教材，也适合于计算机专业水平考试、成人教育、自学考试的人员参考。

图书在版编目（CIP）数据

数据结构教程（C 语言版）/李春葆，金晶编著. 一北京：

清华大学出版社，2006.11

ISBN 7-302-14054-5

Ⅰ. 数…　Ⅱ. ①李…　②金…　Ⅲ. 数据结构

Ⅳ. TP311.12

中国版本图书馆 CIP 数据核字（2006）第 125566 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦

http://www.tup.com.cn　　**邮　　编**：100084

社总机：010-62770175　　**客户服务**：010-82896445

组稿编辑：夏非彼

文稿编辑：张楠

封面设计：林陶

版式设计：科海

印 刷 者：北京市耀华印刷有限公司

发 行 者：新华书店总店北京发行所

开　　本：787×1092　1/16　　**印张**：14.25　　**字数**：347 千字

版　　次：2006 年 10 月第 1 版　　2006 年 10 月第 1 次印刷

书　　号：ISBN 7-302-14054-5/TP・8441

印　　数：1~5000

定　　价：22.00 元

前　言

“数据结构”是计算机专业的一门专业基础课，也是一门核心课程，起到承前启后的作用。学习本课程需要具备一定的C语言的程序设计知识。本书是结合计算机专业的相关大纲编写的，介绍了各种最常用的数据结构，讨论它们在计算机中的存储结构，以及这些数据结构上的操作和实际的算法。

本书力求在阐明各种数据结构的内在逻辑关系、存储结构和相关运算的同时，从编程角度出发，强化各种数据结构运算算法的实现过程，不仅能使读者掌握数据结构涵盖的理论基础知识，更重要的是得到了程序设计的训练和编程能力的提高。

本书共分为8章：第1章是概论部分，讨论数据结构的基本概念和算法描述；第2章是线性表，讨论线性表的逻辑结构、线性表的顺序存储和链式存储；第3章是栈和队列，讨论栈和队列的特点及其各种存储结构与基本操作的实现，并给出了相应的应用实例；第4章是串和数组，讨论串的各种存储结构及其基本操作的实现、数组和稀疏矩阵的各种存储结构及其基本操作的实现；第5章是树和二叉树，讨论树的定义与表示、二叉树的基本操作和哈夫曼树；第6章是图，讨论图的各种存储结构和遍历的实现；第7章是查找，讨论了各种常用的查找方法及其实现；第8章是排序，讨论了各种常用的排序方法及其实现。每章给出一定数量的练习题，供读者选取。最后是两个附录，附录A给出了各章习题的参考答案，附录B给出了本书中C/C++程序的功能索引。

本书具有很强的实用性和可读性，概念原理的阐述力求准确、精炼，写作风格上尽量通俗易懂。对书中的练习题给出了参考答案，便于学生自学。

本书适合于作为计算机及相关专业“数据结构”课程的教材，也适合于计算机水平考试人员参考。

由于水平所限，尽管编者不遗余力，仍可能存在错误和不足之处，敬请读者批评指正。所有代码都可以从www.khp.com.cn网站中免费下载。

编　者

2006.10

目　录

第 1 章 概 论

数据的组织形式和表示方式直接关系到计算机对数据的处理效率，因此为了更好地进行程序设计、有效地利用计算机，就需要对计算机程序加工处理的对象进行系统和深入地研究。研究各种数据的特性以及数据之间存在的关系，进而根据实际应用的要求，合理地组织和存储数据，设计出相应的算法，这就是“数据结构”要讨论的问题。本章介绍数据结构的基本概念和算法分析方法。

1.1 什么是数据结构

数据是对客观事物的符号表示，在计算机科学中是指所有能输入到计算机中并被计算机程序处理的符号的总称，它是计算机程序加工的“原料”。例如，一个班的全部学生记录、a～z 的字母集合、1～1000 间的所有素数等都称为数据。

数据元素（也称为结点）是数据的基本单位，在程序中通常作为一个整体进行考虑和处理。有时一个数据元素可以由若干个数据项组成。数据项是具有独立含义的最小标识单位。如整数这个集合中，10 这个数就可称为一个数据元素。又比如在一个数据库（关系数据库）中，一个记录可称为一个数据元素，而这个元素中的某一字段就是一个数据项。

数据结构是相互之间存在一种或多种特定关系的数据元素的集合。这些数据元素不是孤立存在的，而是有着某种关系，这种关系称为结构。

1.1.1 数据的逻辑结构

数据元素和数据元素之间的逻辑关系称为数据的逻辑结构。根据数据元素之间逻辑关系的不同特性，分为下列 4 类基本结构：

- 集合　结构中的数据元素同属于一个集合（集合类型由于元素之间的关系过于松散，数据结构课程中较少讨论）。
- 线性结构　结构中的数据元素存在一个对一个的关系。
- 树形结构　结构中的数据元素存在一个对多个的关系。
- 图形结构　结构中的数据元素存在多个对多个的关系。

在大多数情况下，数据的逻辑结构可以用二元组来表示：

S=(D,R)

其中 D 是数据结点的有限集合；R 是 D 上的关系的有限集合，其中每个关系都是从 D 到 D 的关系。在表示每个关系时，用尖括号表示有向关系，如<a,b>表示存在结点 a 到结点 b 之间的关系；用圆括号表示无向关系，如(a,b)表示既存在结点 a 到结点 b 之间的关系，又存在结点 b 到结点 a 之间的关系。设 r 是一个 D 到 D 的关系，r∈R，若 d，d'∈K，且<d,d'>∈r，则称 d'是 d 的后继结点，d 是 d'的前驱结点，这时 d 和 d'是相邻的结点（都是相对 r 而言的）；如果不存在一个 d'使<d,d'>∈r，则称 d 为 r 的终端结点；如果不存在一个 d'使<d',d>∈r，则称 d 为 r 的开始结点；如果 d 既不是终端结点也不是开始结点，则称 d 是内部结点。

例如一个城市表，如表 1.1 所示，就可称之为一个数据结构，它由很多记录（这里的数据元素就是记录）组成，每个元素又包括多个字段（数据项）。那么这个表的逻辑结构是怎么样的呢?分析数据结构都是从数据元素之间的关系开始分析，对于这个表中的任一个记录，它只有一个前驱结点和一个后继结点，整个表只有一个开始结点和一个终端结点。当知道了这些关系之后，就能明白这个表的逻辑结构，即为线性结构了。其逻辑结构表示如下：

```
City=(D,R)
D={北京,上海,武汉,西安,南京}
R={r}
r={<北京,上海>,<上海,武汉>,<武汉,西安>,<西安,南京>}
```

表 1.1　城市表

城市	区号	说明
北京	010	首都
上海	021	直辖市
武汉	027	湖北省省会
西安	029	陕西省省会
南京	025	江苏省省会

数据的逻辑结构可以用相应的关系图来表示，称之为逻辑结构图。

【例 1.1】　设数据逻辑结构如下：

```
B1=(D,R)
D={1,2,3,4,5,6,7,8,9}
R={r}
r={<1,2>,<1,3>,<3,4>,<3,5>,<4,6> ,<4,7>,<5,8>,<7,9>}
```

试画出对应的逻辑结构图，并给出哪些是开始结点，哪些是终端结点，说明是何种数据结构。

解：B1 对应的逻辑结构如图 1.1 所示。其中，1 是开始结点；2，6，8，9 是终端结点。它是一种树形结构。

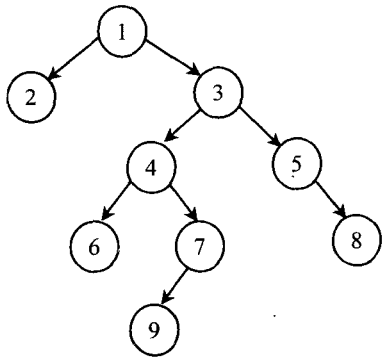

图 1.1　逻辑结构图

1.1.2　数据的存储结构

数据在计算机中的存储表示称为数据的存储结构，也称为物理结构。

把数据存储到计算机中时，通常要求既存储各结点的数值，又存储结点与结点之间的逻辑关系。在实际应用中，数据的存储方法是灵活多样的，可根据问题规模（通常是指结点数目的多少）和运算种类等因素适当选择。本书将介绍 4 种基本的存储结构，它们是：顺序存储结构、链式存储结构、索引存储结构和散列存储结构。下面简要介绍这些存储结构的特点。

1. 顺序存储结构

顺序存储结构是把逻辑上相邻的元素存储在一组连续的存储单元中，其元素之间的逻辑关系由存储单元地址间的关系隐含表示。

对于前面的逻辑结构 City，假定每个元素占用 30 个存储单元，数据从 100 号单元开始由低地址向高地址方向存放，对应的顺序存储结构如图 1.2 所示。

地址	城市名	区号	说明
100	北京	010	首都
130	上海	021	直辖市
160	武汉	027	湖北省省会
190	西安	029	陕西省省会
210	南京	025	江苏省省会

图 1.2　顺序存储结构

顺序存储结构的主要优点是节省存储空间。因为分配给数据的存储单元全用于存放结点的数据值，结点之间的逻辑关系没有占用额外的存储空间。用这种方法来存储线性结构的数据元素时，可实现对各数据元素的随机访问。这是因为线性结构中每个数据元素都对应一个序号（开始结点的序号为 1，它的后继结点的序号为 2，…），可以根据结点的序号 i，计算出结点的存储地址：

$$Loc(i)=q+(i-1)\times p$$

其中，p 是每个元素所占单元数；q 是第一个元素结点所占单元的首地址。

顺序存储结构的主要缺点是不便于修改。在对结点进行插入、删除运算时，可能要移动一系列的结点。

2．链式存储结构

链式存储结构是给每个结点附加指针字段，用于存放相邻结点的存储地址。假定给实例 City 中的每个结点附加一个“下一个结点地址”即后继指针字段，用于存放后继结点的首地址，则可得到如图 1.3 所示的 City 的链式存储表示。图中，每个结点占用两个连续的存储单元，一个存放结点的数值，另一个存放后继结点的首地址。

地址	城市名	区号	说明	下一个结点地址
100	北京	010	首都	210
130	南京	025	江苏省省会	∧
160	西安	029	陕西省省会	130
190	武汉	027	湖北省省会	160
210	上海	021	直辖市	190

图 1.3　链式存储结构

链式存储结构的主要优点是便于修改，在进行插入、删除运算时，仅需修改结点的指针字段值，不必移动结点。

与顺序存储结构相比，链式存储结构的主要缺点是存储空间的利用率较低，因为分配给数据的存储单元有一部分被用来存放结点之间的逻辑关系了。另外，由于逻辑上相邻的结点在存储器中不一定相邻，因此，在用这种方法存储的线性结构上不能对结点进行随机访问。

3．索引存储结构

索引存储结构是在存储结点信息的同时，还建立附加的索引表。索引表中的每一项称为索引项，索引项的一般形式是：(关键字,地址)。关键字惟一标识一个结点；地址作为指向结点的指针。对于线性结构来说，各结点的地址在索引表中是按结点的序号依次排列的。图 1.4 所示是 City 的一种索引存储表示。

在进行关键字查找时，可以先在索引表中快速查找（因为索引表中按关键字有序排列，可以采用二分查找）到相应的关键字，然后通过地址找到结点表中对应的记录。

线性结构采用索引存储后，可以对结点进行随机访问。在进行插入、删除运算时，由于只需移动存储在索引表中的结点的存储地址，而不必移动存储在结点表中的结点的数值，所以仍可保持较高的运算效率（这是因为在一般情况下结点中总包含有多个字段，移动一个结点的数值要比移动一个结点的地址花费更多的时间）。

索引存储结构的缺点是，为建立索引表而增加了时间和空间的开销。

（a）索引表

地址	关键字	指针
300	010	100
310	021	130
320	025	210
330	027	160
340	029	190

（b）结点表

地址	城市名	区号	说明
100	北京	010	首都
130	上海	021	直辖市
160	武汉	027	湖北省省会
190	西安	029	陕西省省会
210	南京	025	江苏省省会

图 1.4　索引存储结构

4．散列存储结构

散列存储结构是根据结点的值确定结点的存储地址。具体做法是：以结点中某个字段的值为自变量，通过某个函数（称为散列函数）计算出对应的函数值 i，再把 i 当作结点的存储地址。

对于 City 结构，假设以城市名的值作为自变量 key，选用函数：

H(key)=ASC(LEFT(Hz(key),1)) mod 8

来计算结点的存储地址，其中，Hz(x)表示提取汉字 x 的拼音串，ASC(LEFT(Hz(key),1))表示取汉字串 key 中第一个拼音字符的 ASCII 码，mod 是取模运算，计算结果如下：

key	北京	上海	武汉	西安	南京
ASC(LEFT(Hz(key),1))	66	83	87	88	78
H(key)	2	3	7	0	6

于是，可以得到图 1.5 所示 City 的散列存储表示。

地址	城市名	区号	说明
0	西安	029	陕西省省会
1			
2	北京	010	首都
3	上海	021	直辖市
4			
5			
6	南京	025	江苏省省会
7	武汉	027	湖北省省会

图 1.5　散列存储结构

散列存储的优点是查找速度快，只要给出待查找结点的数值，就有可能立即算出结点的存储地址。

与前 3 种存储方法不同的是，散列存储方法只存储结点的数值，不存储结点与结点之间的逻辑关系。散列存储方法一般只用于要求对数据能够进行快速查找、插入的场合。采用散列存储的关键是要选择一个好的散列函数和处理“冲突”的办法。本书第 7 章将详细介绍这种存储方法。

在用高级语言编程时，可以用编程语言所提供的数据类型来描述数据的存储结构。例如，用“一维数组”表示一组连续的存储单元，来实现顺序存储结构、索引存储结构和散列存储结构；用 C/C++语言中的“指针”来实现链式存储结构。

1.1.3 数据的运算

数据运算就是施加于数据的操作。比如，有一张表格，需要对它进行查找、增加、修改、删除记录等工作。在数据结构中，运算不仅仅是加、减、乘、除这些算术运算，还常常涉及算法问题。算法的实现是与数据的存储结构密切相关的。

1.1.4 数据结构和数据类型

将按某种逻辑关系组织起来的一组数据元素，按一定的存储方式存储于计算机中，并在其上定义了一个运算的集合，称为一个数据结构。

而数据类型是高级程序设计语言中的一个基本概念，它和数据结构的概念密切相关。

一方面，在程序设计语言中，每一个数据都属于某种数据类型。类型明显或隐含地规定了数据的取值范围、存储方式以及允许进行的运算。因此可以认为，数据类型是在程序设计语言中已经实现了的数据结构。

另一方面，在程序设计过程中，当需要引入某种新的数据结构时，必须借助编程语言所提供的数据类型来描述数据的存储结构。

下面总结 C/C++语言中常用的数据类型。

（1）C/C++语言的基本数据类型

C/C++语言中的基本数据类型有 int 型、float 型和 char 型。int 型可以有 3 个修饰符：short（短整数）、long（长整数）和 unsigned（无符号整数）。

（2）C/C++语言的指针类型

C/C++语言允许直接对存放变量的地址进行操作。如定义：

```
int i,*p;
```

其中，i 是整型变量，p 是指针变量（它用于存放某个整型变量的地址）。表达式&i 表示变量 i 的地址，将 p 指向整型变量 i 的运算为：p=&i。

对于指针变量 p，表达式*p 是取 p 所指变量的值，例如：

```
i=2;p=&i;
printf("%d\n",*p);
```

输出结果为 i 的值即 2。可以使用 malloc()函数为指针分配一片连续的空间，例如：

```
char *p;
```

```
p=(char *)malloc(10*sizeof(char));
p="China";
printf("%c\n",*p);
printf("%s\n",p);
```

上述代码先定义字符指针变量p（它没有指向有效的字符数据，即p的值没有意义），使用malloc()函数为其分配长度为10个字符的空间，将该空间的首地址赋给p（尽管不知道这个地址是多少，但p的值已经有意义了），再将字符串“China”放到这个空间中，所以第1个printf语句输出的是首地址的字符即“C”，而第2个printf语句输出的是整个字符串即“China”。

（3）C/C++语言的数组类型

数组是同一类型的一组有序数据的集合。数组有一维数组和多维数组。数组名标识一个数组，下标指示一个数组元素在该数组中的顺序位置。

数组下标的最小值称为下界，在C/C++语言中总是为0。数组下标的最大值称为上界，在C/C++语言中数组上界为数组定义值减1。例如，int a[10]定义了包含10个整数的数组a。

（4）C/C++语言中的结构体类型

结构体是由一组称为结构体成员的数据项组成的，每个结构体成员都有自己的标识符。例如：

```
struct teacher
{   int no;
    char name[8];
    int age;
};
```

定义了一个结构体类型teacher，有以下代码：

```
struct teacher t1={1,"陈华",34};                    /*定义结构体变量t1并初始化*/
struct teacher t2={5,"王强",48};                    /*定义结构体变量t2并初始化*/
printf("%d,%s,%d\n",t1.no,t1.name,t1.age); /*输出结构体变量t1的各成员值*/
printf("%d,%s,%d\n",t2.no,t2.name,t2.age); /*输出结构体变量t2的各成员值*/
```

执行结果如下：

```
1,陈华,34
5,王强,48
```

（5）C/C++语言中的共用体类型

共用体是把不同的成员组织为一个整体，它们在存储器中共享一段存储单元，但不同成员以不同的方式被解释。例如：

```
union tag
{   short int n;
    char ch[2];
};
```

定义了一个共用体类型 tag，有以下代码：

```
union tag u;
u.n=0x4142;
printf("%c,%c\n",u.ch[1],u.ch[0]);
```

通过赋值后，成员 n 的两个字节中，高位为 64，低位为 65，分别对应 u.ch[1]和 u.ch[0]成员，所以 printf 输出为 A 和 B。

（6）C/C++语言中的自定义类型

C/C++语言中允许使用 typedef 关键字来定义同名的数据类型名，例如：

```
typedef char ElemType;
```

将 char 类型与 ElemType 等同起来。特别是将代码较长的结构体类型定义用新类型来替代，例如：

```
typedef struct student
{   int no;
    char name[10];
    char sex;
    int cno;
} StudType;
```

这样，StudType 等同于学生结构体类型，可以使用该类型定义变量：

```
StudType s1,s2;
```

等同于：

```
struct student s1,s2;
```

1.2 算法和算法分析

1.2.1 算法及其特征

算法是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每条指令表示一个或多个操作。算法有以下 5 个重要特征：

- 有穷性　一个算法必须总是（对任何合法的输入值）在执行有穷步之后结束，且每一步都可在有穷时间内完成。
- 确定性　算法中每一条指令必须有确切的含义，不会产生二义性。
- 可行性　一个算法是可行的，即算法中描述的操作都是可以通过已经实现的基本运算的有限次执行来实现。
- 输入性　一个算法有零个或多个的输入。
- 输出性　一个算法有一个或多个的输出。

【例 1.2】　考虑下列两段描述：

```
(1) void exam1()
    {
        n=2;
        while (n%2==0)
            n=n+2;
        printf("%d\n",n);
    }
```

```
(2) void exam2()
    {
        y=0;
        x=5/y;
        printf("%d,%d\n",x,y);
    }
```

这两段描述均不能满足算法的特征，试问它们违反了哪些特征？

解：（1）是一个死循环，违反了算法的有穷性特征。（2）包含除零错误，违反了算法的可行性特征。

1.2.2 算法描述

描述算法的方法很多，有的采用类 PASCAL，有的采用自然语言等。本书采用 C/C++ 语言来描述算法的实现过程。下面总结常用的用于描述算法的 C 语言语句。

（1）输入语句

```
scanf(格式控制字符串,输入项表);
```

（2）输出语句

```
printf(格式控制字符串,输出项表);
```

（3）赋值语句

```
变量名=表达式;
```

（4）条件语句

```
if <条件> <语句>;
```

或者

```
if <条件> <语句1> else <语句2>;
```

（5）循环语句

- while 循环语句

```
while 表达式
    循环体语句;
```

- do-while 循环语句

```
do
    循环体语句;
while 表达式;
```

- for 循环语句