数据结构

张晓静　主编

海洋出版社

数 据 结 构

张晓静　主编

海洋出版社

2004 年 · 北京

内 容 简 介

数据结构是计算机专业教学计划中的一门核心课程，也是信息管理、通信电子等与计算机技术关系密切的专业的一门基础课程。本书对C++语言作了简单介绍，叙述了抽象数据类型，介绍了线性表、栈、队列、数组、广义表、树和图等数据结构，并且介绍了查找和排序方法。对大多数算法和算法的应用给出了相应的C++程序，每章后面附有大量的习题，便于教学。

本书是为高等院校开设“数据结构”课程编写的教材，可作为计算机专业及相关专业本科、专科学生的教材使用，也可供从事计算机软件开发和应用的工程技术人员阅读。

图书在版编目(CIP)数据

数据结构 / 张晓静主编. —北京：海洋出版社，2004.1
ISBN 7-5027-6074-1

Ⅰ.数… Ⅱ.张… Ⅲ.数据结构-高等学校-教材 Ⅳ.TP311.12

中国版本图书馆CIP数据核字(2004)第004145号

责任编辑：高显刚
责任印制：严国晋

海洋出版社 出版发行
http://www.oceanpress.com.cn
(100081 北京市海淀区大慧寺路8号)
北京蓝空印刷厂印刷 新华书店发行所经销
2004年1月第1版 2004年1月北京第1次印刷
开本：787mm×1092mm 1/16 印张：18.25
字数：400千字 印数：1～4500册
定价：30.00元

《数据结构》编委会

前　言

数据结构是计算机专业教学计划中的一门核心课程，也是信息管理、通信电子、自动控制等与计算机技术关系密切的专业的一门基础课程。要从事与计算机科学与技术相关的工作，尤其是计算机应用领域的开发和研制工作，必须具备坚实的数据结构的基础。

数据结构课程的教学目的是使学生学会分析研究计算机所要加工处理的数据的特征，掌握组织数据、存储数据和处理数据的基本方法，并加强在实际应用中选择合适的数据结构和相应算法的训练。

数据结构课程内容丰富，学习量大，贯穿于全书的动态链表存储结构和递归技术令不少初学者望而生畏。作者长期从事数据课程的教学，对该课程的教学特点和难点有比较深切的体会。在认真总结多年教授“数据结构”课程的基础上，吸收国内外各种数据结构教材的优点，对多年来形成的数据结构课程的教学内容进行了合理的编排，既强调数据结构的原理和方法，又注重其实践性，以适应现代大学生的学习特点和要求。

本书的一个重要特点就是将程序设计和数据结构尽可能结合起来，在书的最后给出了几个完整的C++程序，使学生对C++有比较全面和深入的了解，并且便于上机，从而为数据结构课程的实验建立良好的基础。

全书共分为9章和一个附录，第1章介绍数据结构、算法及其度的基本概念，对C++作了简单介绍，并叙述了抽象数据类型；第2～5章介绍了线性结构——线性表、栈、队列、串、数组、广义表；第6章和第7章分别介绍了非线性结构——树和图；第8章和第9章分别介绍了查找和排序的方法；第10章介绍了文件操作；附录给出了线性表、栈、树等基本操作C++完整程序。

本书第1～3章由张晓静、张宗镛编写，第4、5章由张娜萍编写，第6章由张晓静、张雅洁编写，第7章由张娜萍、米燕涛编写，第8、9章由白云飞、倪素虹编写，附录由程欣、王巧玲、齐林、刘伟编写，全书由张晓静统稿，张晓静、张宗镛校对。

由于时间仓促、作者水平有限，书中难免存在一些缺点和错误，殷切希望广大读者批评指正。

编者

2003年10月

目　次

第1章 绪 论

本章主要介绍数据结构课程中的一些常用术语的含义，集合、线性结构、树和图等常用数据结构的表示，C++语言中一些常用系统头文件的作用，函数重载、运算符重载和数据文件的使用，算法的时间复杂度和空间复杂度评价方法等内容。

1.1 基本术语

在本节中，我们将对一些概念和术语赋以确定的含义，这些概念和术语将在以后的章节中多次出现。

数据(Data) 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”。如一个人的名字可以用一个字符串来描述，一条曲线可以用一个数组来描述，数组中的每一个元素用来存储曲线中对应点的坐标值和颜色编号。因此，一个文档、记录、数组、句子、单词、算式、符号、声音、图形、图像等都统称为数据。

数据元素(Data Element) 简称元素，它是一个数据整体中相对独立的单位。如对于一个文件来说，每个记录就是它的数据元素；对于一个字符串来说，每个字符就是它的数据元素；对于一个数组来说，每一个数值就是它的数据元素。数据和数据元素是相对而言的。如对于一个记录来说，它是所属文件的一个数据元素，而相对于所含的数据项而言，它又是数据，数据项是它的一个元素。

数据记录(Data Record) 简称记录，它是数据处理领域组织数据的基本单位。数据中的每个数据元素在许多应用场合被组织成记录的结构。一个数据记录由一个或多个数据项所组成。例如图书管理目录，每个记录表示一本图书的目录信息，如表1-1所示。

表1-1 图书目录表

登录号	书 名	作 者	出版者
001	C++语言及其程序设计基础	张国峰	电子工业出版社
002	操作系统	孟庆昌	中央电视大学出版社
003	新编VB程序设计教程	梁普选	电子工业出版社
004	信息技术基础	刘明生	河北大学出版社
005	数据结构	严蔚敏	清华大学出版社
⋮	⋮	⋮	⋮

在表 1－1 中，第一行为目录行，给出了该表中每条记录的结构。从第二行开始向下的每一行为一条记录，它给出了一本图书的有关信息；每一列为一个数据项，它描述了图书中的一种属性。

在一个表中，若所有记录的某个数据项的值都不同，即，每个值能够惟一地标识一个记录时，则可把这个数据项作为记录的关键数据项，简称关键项。关键项中的每一个值称为所在记录的关键字。在表 1－1 中，登录号数据项的值都不同，所以可把登录号作为记录的关键项，其中的每一个值就是所在记录的关键字。

数据结构(Data Structure)是指数据及其相互之间的联系。数据之间的相互联系，被称为数据的逻辑结构。在计算机中存储数据时，不仅要存储数据本身，而且要存储它们之间的联系(即逻辑结构)。一种数据结构在存储器中的存储方式称为数据的物理结构或存储结构。由于存储方式有顺序、链接、索引和散列等多种形式，所以，一种数据结构可以根据应用的需要表示成一种或几种存储结构。

数据的逻辑结构和存储结构都反映数据的结构，但通常所说的数据结构是指数据的逻辑结构。它一般包括以下三个方面的内容：

(1)逻辑结构：数据元素之间的逻辑关系。

(2)物理结构或存储结构：数据元素及其关系在计算机存储器内的表示。

(3)运算：对数据所施加的操作。

通常数据结构(逻辑结构)采用二元组表示：

$B=(K,R)$

B 是一种数据结构，它由数据元素的集合 K 和 K 上关系集合 R 所组成。其中

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

$R=\{<k_u,k_v> \mid k_u,k_v\in K\}$

其中 k_i 表示集合 K 中的一个数据元素，n 为 K 中数据元素个数，若 $n=0$，则 K 是一个空集。对于较复杂的数据结构，可能包含有几个二元关系。

K 上的关系集 R 是序偶的集合。对于 R 中的任一序偶 $<x,y>(x,y\in K)$，我们把 x 叫做序偶的第一个元素或前驱，把 y 叫做序偶的第二个元素或后继。

下面介绍几种典型的数据结构。

例 1 一种数据结构 $set=(K,R)$，其中

$K=\{1,2,3,4,5,6,7,8\}$

$R=\{\}$

在该数据结构中，只存在元素的集合，而关系集合 R 为空集。具有这种特点的数据结构被称为集合结构。对于集合结构，可以看做元素按任一次序排列的线性结构，在存储器中可以根据需要按任一种存储方式存储。

例 2 一种数据结构 $L=(K,R)$

$K=\{1,2,3,4,5,6,7,8\}$

$R=\{\langle 5,1\rangle,\langle 1,3\rangle,\langle 3,2\rangle,\langle 2,4\rangle,\langle 4,7\rangle,\langle 7,8\rangle,\langle 8,6\rangle\}$

对应的图形如图 1－1 所示。

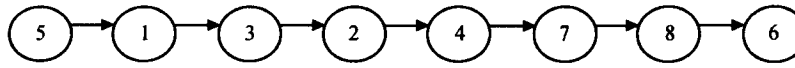

图 1-1 数据的线性结构示意图

在 L 中,每个元素有且仅有一个直接前驱元素(除第一个元素 5 外),有且仅有一个直接后继元素(除最后一个元素 6 外)。这种数据结构的特点是数据元素之间的 1 对 1(1:1)联系,即线性关系,我们把具有这种特点的数据结构称为线性结构。

例 3 一种数据结构 T=(K,R) 其中

K={1,2,3,4,5,6,7,8}

R={⟨1,2⟩,⟨1,3⟩,⟨1,4⟩,⟨2,5⟩,⟨2,6⟩,⟨3,7⟩,⟨3,8⟩}

对应的示意图如图 1-2 所示。

图 1-2 像一棵倒画的树,最上面的一个没有前驱只有后继的结点叫做树根结点,最下面一层的只有前驱没有后继的结点叫做树叶结点,除树根结点和树叶结点之外的结点叫做树枝结点。在一棵树中,每个结点有且只有一个前驱结点(除树根结点外),但可以有任意多个后继结点(树叶结点可以看做具有 0 个后继结点)。这种数据结构的特点是数据元素之间的 1 对 N(1:N)联系(N≥0),即层次关系。我们把具有这种特点的数据结构称为树结构,简称树。

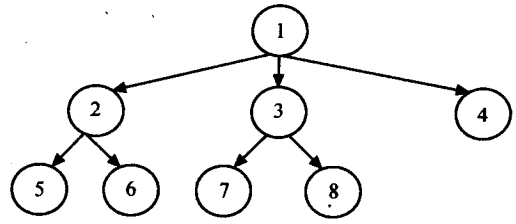

图 1-2 数据的树结构示意图

例 4 一种数据结构 G=(K,R),其中

K={1,2,3,4,5,6,7}

R={(1,2),(1,4),(2,3),(2,6),(2,7),(3,7),(4,6),(5,7)}

对应的图形如图 1-3 所示。

从图 1-3 可以看出,结点之间的联系是 M 对 N(M:N)联系(M≥0,N≥0),即网状关系。也就是说,每个结点之间可以有任意个前驱结点和任意个后继结点。我们把具有这种特点的数据结构称为图结构,简称图。

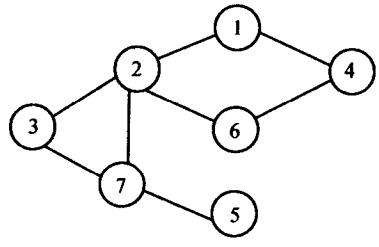

图 1-3 数据的图形结构示意图

从图结构、树结构和线性结构的定义可知,树结构是图形结构的特殊情况,线性结构是树形结构的特殊情况。为了区别于线性结构,我们把树结构和图结构统称为非线性结构。

数据类型(Data Type)是和数据结构密切相关的一个概念,它最早出现在高级程序语言中,用以刻画(程序)操作对象的特性。在用高级程序语言编写的程序中,每个变量、常量或表达式都有一个它所属的确定的数据类型。类型明显或隐含地规定了在程序执行期间变量或表达式所有可能取值的范围以及在这些值上允许进行的操作。因此数据类型是

一个值的集合和定义在这个值集上的一组操作的总称。

数据类型可分为简单类型和结构类型两种。简单类型中的每个数据(即简单数据)通常只作为整体使用,如一个整数、实数、字符、指针、枚举量等都是这样的、不可分割使用的整体数据。结构类型由简单类型按照一定的规则构造而成,并且结构类型仍可以包含结构类型,所以一种结构类型汇总的数据(即结构数据)可以分解为若干简单数据或结构数据,每个结构数据仍可再分。如数组是一种结构类型,它由固定个数的同一类型顺序排列而成。记录也是一种结构类型,它由固定个数的不同(也可以相同)类型顺序排列而成。另外文件和字符串也是一种结构类型。

数据对象(Data Object)简称对象,是性质相同的数据元素的集合,是数据的一个子集,它属于一种数据类型的特定实例。例如 30 为一个整形数据对象,'A'为一个字符数据对象,int a[10]定义 a 为一个含有 10 个整型数的数组对象。

算法(Algorithm)是对特定问题求解步骤的一种描述,它是指令的有限序列,其中每一条指令表示一个或多个操作。作为一个算法应具备以下 5 个特性:

(1) 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。

(2) 确定性　算法中每一条指令必须有确切的含义,使得理解时不会产生二义性。并且在任何条件下,算法只有惟一的一条执行路径,即对于相同的输入只能得出相同的输出。

(3) 可行性　算法中的每一步都必须是可行的,即每一步都能够通过手工或机器接受任何有限次操作,在有限时间内实现。

(4) 输入　一个算法可以有零个、一个或多个输入量,在算法被执行之前提供给算法。

(5) 输出　一个算法执行结束后至少有一个输出量,它是利用算法对输入量进行运算和处理的结果。

1.2　C++语言基础

C++与C具有许多相同的功能,但除了数据抽象和继承以外,C++对C有很多扩充的功能。我们认为本书读者已经熟悉C语言。

1.2.1　程序结构

一个C++程序可由若干个文件组成。C++的文件分为头文件和源文件两类。

头文件以.h为后缀,用于存放函数声明,它给出了函数的参数类型、个数以及函数的返回类型,称为原型。有一些头文件是系统定义的,如<iostream.h>,而还有一些头文件是用户定义的;而源文件用来存放C++的源代码,其后缀为.cpp。可通过预处理指令#include将头文件包含在适当的文件中。

例5　下面是一个模拟数字式时钟的程序,它在屏幕上打印时、分、秒,程序被存于几个文件中。

```
//clock.h
struct clock{
   int hour, minute, second;
};
void display(clock * );
void update(clock * );
//clock.cpp
#include <iostream.h>
#include "clock.h"
void update(clock * t)
{  t->second++;
   if (t->second==60) {
         t->second=0;
         t->minute++;
       }
       if (t->minute==60) {
         t->minute=0;
          t->hour++;
        }
        if (t->hour==24)
           t->hour=0;
}
void display (clock * t)
{
       cout<<t->hour<<':'<<t->minute<<':'<<t->second<<endl;
}
//main.cpp
# include "clock.h"
void delay;
main()
{
clock aClock;
aClock.hour=aClock.minute=aClock.second=0;
for(;;){
       update (&aClock);
       display (&aClock);
       delay();
      }
```

```
}
void delay()
{
  for (int t=0;t<12800; t++)
    ;
}
```

在本例子中,头文件 clock .h 定义了结构类型 clock,在文件 clock .cpp 中定义了用于操作该类型的变量的两个函数。主程序 main()用于模拟一个时钟,程序中函数 delay()用于延迟一段时间。

将函数和数据定义在互相独立的源文件中,可支持模块化的程序设计。在各个源文件使用后缀为".H"的头文件,定义了对其他各个模块的调用接口。

1.2.2 常用包含文件

包含文件语句是以关键字 # include 开头,后跟用尖括号或双引号括起来的头文件名,行后不需要使用分号。下面介绍几个常用的系统头文件的使用。

(1) # include <iostream.h>

在程序的开始使用该命令后,在其后的每一个函数中,都可以使用标准输入设备(键盘)流对象 cin,标准输出设备(屏幕)流对象 cout 和标准错误输出设备(屏幕)对象 cerr,以及使用用于输入的提取操作符">>"和用于输出的插入操作符"<<"进行数据输入输出操作。对于基本类型为 char, short, int, long, float, double, long double 的数据能够直接进行输入和输出;对于字符指针类型(char *)的数据能够输出指针所指向的字符串;对于非字符指针类型的指针数据能够直接输出(即操作数地址);对于用户定义类型数据,只有通过对">>"和"<<"操作符重载后才能进行整体输入和输出。例如,有 int a 和 char b,可以使用如下输入输出语句进行输入输出操作:

```
cin>>a>>b;
cout<<a<<"    "<<b;
```

另外,在使用 # include <iostream.h>语句之后的函数中,允许使用换行符常量 endl 和空指针常量 NULL,它们分别表示换行符'\n'和数值 0。

(2) # include <stdlib.h>

在 stdio.h 头文件中含有 void exit(int), int rand (void), void srand(unsigned)等函数的原型。exit (int)函数的作用是结束程序的执行。一般用整数值 0 调用该函数表示正常结束,用整数值 1 调用该函数表示非正常结束。rand( )函数的作用是返回 0~32767 之间的一个随机整数。srand (unsigned)函数的作用是初始化随机数发生器,当参数不同时,由 rand()函数所产生的随机数序列也不同。

(3) # include <fstream.h>

fstream.h 为使用文件流类的头文件,其中定义有输入文件流类 ifstream,输出文件流类 ofstream 和输入输出文件流类 fstream。利用它们可以为编程者定义相应的文件流对象,从而对外存上的文件进行输入输出操作。例如:

```
ifstream input("wj1.dat", ios::in|ios::nocreate);
ofstream output1("wj2.dat", ios::out);
ofstream output2("wj3.dat", ios::app);
fstream inout("wj4.dat", ios::in|ios::out);
```

在以上每一条语句中都定义了一个相应的文件流对象,其后括号内的第一个参数给出要打开的实际文件,它为一个字符指针类型,第二个参数给出文件的打开方式。执行上述任一条语句后,若相应的文件被打开,则由文件流对象返回一个非0值,否则返回一个0值。当打开一个文件后,将在内存中开辟一个相应的文件缓冲区,通过文件流对象访问缓冲区,实现对文件的读写操作。

第一条语句定义了输入文件流对象为input,要打开的文件为当前目录下的"wj1.dat",并由ios::in参数规定按输入(即文件到内存)方式打开,此参数可以缺省,由ios::nocreate参数规定,若指定文件不存在,则不应去建立它,否则将建立它。打开一个用于输入的文件后,文件指针被自动指向文件内容的开始位置。

第二条语句定义了一个输出文件流对象为output1,要打开的文件为当前目录下的"wj2.dat",并由ios::out参数规定按输出(即由内存到文件)方式打开,此参数可以缺省,若指定的文件不存在,则自动在当前目录下建立文件名"wj2.dat"的空文件。打开一个用于输出的文件后,文件中的原有内容自动被清除。

第三条语句定义了输出文件流对象为output2,要打开的文件为当前目录下的"wj3.dat",并由ios::app参数规定按追加方式打开,同样若文件不存在则自动建立它。打开一个用于追加输出的文件后,文件中的原有内容保持不变,文件指针自动移到文件的末尾。

第四条语句定义了输入输出文件流对象为inout,对应的文件为"wj4.dat",并规定既可对该文件进行输入操作,也可对该文件进行输出操作。打开一个同时用于输入和输出的文件后,文件中的原有内容不变,文件指针被自动移到文件内容的开始位置。

(4) # include <string.h>

string.h为进行字符串操作的头文件函数,其中定义有一些字符串函数的原型。用户可以在程序中直接调用这些函数处理字符串。常用的字符串函数有:

1) int strlen (const char * s);

返回s指针所指字符串的长度,字符串的空结束符('\0')不计算在内。

2) char * strcpy (char * dest, const char * src)

把src所指字符串拷贝到dest指针所指的存储空间中,该函数返回dest指针。

3) char * strcat (char * dest, const char * src);

把src所指字符串拷贝到dest所指字符串后面的存储空间中,连接后dest所指字符串的长度等于dest原有长度和src串长度之和,该函数返回dest指针。

4) int strcmp (const char * s1, const char * s2);

把s1所指字符串同s2所指字符串比较,若s1串大于s2串则返回值大于0,若s1串小于s2串则返回值小于0,若s1串等于s2串则返回值等于0。

5) char * strchr(const char * s, int c);

从s所指字符串的开始顺序查找ASCII码为c值的字符,若查找成功则返回指向该