21世纪

高等院校规划教材

蔡明志　编著

数据结构
——用C语言描述

注重学科体系的完整性，兼顾考研学生需要
强调理论与实践相结合，注重培养专业技能

中国水利水电出版社
www.waterpub.com.cn

21世纪高等院校规划教材

数据结构——用C语言描述

蔡明志　编著

中国水利水电出版社

内 容 提 要

本书是根据作者多年教学的经验，并参考了近年出版的多种国外大学数据结构教科书而编写的。

本书以C语言为程序设计语言，采用系列式的叙述方式，引导读者循序渐进地掌握数组、链接表、栈和队列、树与森林、图和堆等不同的数据结构，并系统地介绍了查找和排序的各种实现方法。对每一种数据结构，除了详细阐述其基本概念和具体实现外，都尽可能地对每种操作给出C语言的算法描述；对查找和排序的各种算法，还着重在时间上做出定量或定性的分析比较。

本书不但涉及内容广泛、涵盖的知识点全面，而且条理清晰、通俗易懂、图文并茂，有利于学生系统性地学习。

本书可作为计算机类专业或软件专业的本科或专科教材，也可供从事计算机工程与应用工作的科技工作者参考。

本书配有电子教案，读者可到中国水利水电出版社网站上下载，网址为：http://www.waterpub.com.cn/softdown。

图书在版编目(CIP)数据

数据结构：用C语言描述／蔡明志编著．－北京：中国水利水电出版社，2005

（21世纪高等院校规划教材）

ISBN 7-5084-3428-5

Ⅰ．数… Ⅱ．蔡… 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．①TP311.12②TP312

中国版本图书馆CIP数据核字（2005）第139160号

书　　名	**数据结构——用C语言描述**
作　　者	蔡明志　编著
出版发行	中国水利水电出版社（北京市三里河路6号　100044） 网址：www.waterpub.com.cn E-mail：mchannel@263.net（万水） sales@waterpub.com.cn 电话：（010）63202266（总机）、68331835（营销中心）、82562819（万水）
经　　售	全国各地新华书店和相关出版物销售网点
排　　版	北京万水电子信息有限公司
印　　刷	北京市天竺颖华印刷厂
规　　格	787mm×1092mm　16开本　24.5印张　598千字
版　　次	2006年1月第1版　2006年1月第1次印刷
印　　数	0001—4000册
定　　价	34.00元

序

随着计算机科学与技术的飞速发展，计算机的应用已经渗透到国民经济与人们生活的各个角落，正在日益改变着传统的人类工作方式和生活方式。在我国高等教育逐步实现大众化后，越来越多的高等院校会面向国民经济发展的第一线，为行业、企业培养各级各类高级应用型专门人才。为了大力推广计算机应用技术，更好地适应当前我国高等教育的跨跃式发展，满足我国高等院校从精英教育向大众化教育的转变，符合社会对高等院校应用型人才培养的各类要求，我们成立了“21 世纪高等院校规划教材编委会”，在明确了高等院校应用型人才培养模式、培养目标、教学内容和课程体系的框架下，组织编写了本套“21 世纪高等院校规划教材”。

众所周知，教材建设作为保证和提高教学质量的重要支柱及基础，作为体现教学内容和教学方法的知识载体，在当前培养应用型人才中的作用是显而易见的。探索和建设适应新世纪我国高等院校应用型人才培养体系需要的配套教材已经成为当前我国高等院校教学改革和教材建设工作面临的紧迫任务。因此，编委会经过大量的前期调研和策划，在广泛了解各高等院校的教学现状、市场需求，探讨课程设置、研究课程体系的基础上，组织一批具备较高的学术水平、丰富的教学经验、较强的工程实践能力的学术带头人、科研人员和主要从事该课程教学的骨干教师编写出一批有特色、适用性强的计算机类公共基础课、技术基础课、专业及应用技术课的教材以及相应的教学辅导书，以满足目前高等院校应用型人才培养的需要。本套教材消化和吸收了多年来已有的应用型人才培养的探索与实践成果，紧密结合经济全球化时代高等院校应用型人才培养工作的实际需要，努力实践，大胆创新。教材编写采用整体规划、分步实施、滚动立项的方式，分期分批地启动编写计划，编写大纲的确定以及教材风格的定位均经过编委会多次认真讨论，以确保该套教材的高质量和实用性。

教材编委会分析研究了应用型人才与研究型人才在培养目标、课程体系和内容编排上的区别，分别提出了 3 个层面上的要求：在专业基础类课程层面上，既要保持学科体系的完整性，使学生打下较为扎实的专业基础，为后续课程的学习做好铺垫，更要突出应用特色，理论联系实际，并与工程实践相结合，适当压缩过多过深的公式推导与原理性分析，兼顾考研学生的需要，以原理和公式结论的应用为突破口，注重它们的应用环境和方法；在程序设计类课程层面上，把握程序设计方法和思路，注重程序设计实践训练，引入典型的程序设计案例，将程序设计类课程的学习融入案例的研究和解决过程中，以学生实际编程解决问题的能力为突破口，注重程序设计算法的实现；在专业技术应用层面上，积极引入工程案例，以培养学生解决工程实际问题的能力为突破口，加大实践教学内容的比重，增加新技术、新知识、新工艺的内容。

本套规划教材的编写原则是：

在编写中重视基础，循序渐进，内容精炼，重点突出，融入学科方法论内容和科学理念，反映计算机技术发展要求，倡导理论联系实际和科学的思想方法，体现一级学科知识组织的层次结构。主要表现在：以计算机学科的科学体系为依托，明确目标定位，分类组织实施，兼容互补；理论与实践并重，强调理论与实践相结合，突出学科发展特点，体现

学科发展的内在规律；教材内容循序渐进，保证学术深度，减少知识重复，前后相互呼应，内容编排合理，整体结构完整；采取自顶向下设计方法，内涵发展优先，突出学科方法论，强调知识体系可扩展的原则。

本套规划教材的主要特点是：

（1）面向应用型高等院校，在保证学科体系完整的基础上不过度强调理论的深度和难度，注重应用型人才的专业技能和工程实用技术的培养。在课程体系方面打破传统的研究型人才培养体系，根据社会经济发展对行业、企业的工程技术需要，建立新的课程体系，并在教材中反映出来。

（2）教材的理论知识包括了高等院校学生必须具备的科学、工程、技术等方面的要求，知识点不要求大而全，但一定要讲透，使学生真正掌握。同时注重理论知识与实践相结合，使学生通过实践深化对理论的理解，学会并掌握理论方法的实际运用。

（3）在教材中加大能力训练部分的比重，使学生比较熟练地应用计算机知识和技术解决实际问题，既注重培养学生分析问题的能力，也注重培养学生思考问题、解决问题的能力。

（4）教材采用“任务驱动”的编写方式，以实际问题引出相关原理和概念，在讲述实例的过程中将本章的知识点融入，通过分析归纳，介绍解决工程实际问题的思想和方法，然后进行概括总结，使教材内容层次清晰，脉络分明，可读性、可操作性强。同时，引入案例教学和启发式教学方法，便于激发学习兴趣。

（5）教材在内容编排上，力求由浅入深，循序渐进，举一反三，突出重点，通俗易懂。采用模块化结构，兼顾不同层次的需求，在具体授课时可根据各校的教学计划在内容上适当加以取舍。此外还注重了配套教材的编写，如课程学习辅导、实验指导、综合实训、课程设计指导等，注重多媒体的教学方式以及配套课件的制作。

（6）大部分教材配有电子教案，以使教材向多元化、多媒体化发展，满足广大教师进行多媒体教学的需要。电子教案用 PowerPoint 制作，教师可根据授课情况任意修改。相关教案的具体情况请到中国水利水电出版社网站 www.waterpub.com.cn 下载。此外还提供相关教材中所有程序的源代码，方便教师直接切换到系统环境中教学，提高教学效果。

总之，本套规划教材凝聚了众多长期在教学、科研一线工作的教师及科研人员的教学科研经验和智慧，内容新颖，结构完整，概念清晰，深入浅出，通俗易懂，可读性、可操作性和实用性强。本套规划教材适用于应用型高等院校各专业，也可作为本科院校举办的应用技术专业的课程教材，此外还可作为职业技术学院和民办高校、成人教育的教材以及从事工程应用的技术人员的自学参考资料。

我们感谢该套规划教材的各位作者为教材的出版所做出的贡献，也感谢中国水利水电出版社为选题、立项、编审所做出的努力。我们相信，随着我国高等教育的不断发展和高校教学改革的不断深入，具有示范性并适应应用型人才培养的精品课程教材必将进一步促进我国高等院校教学质量的提高。

我们期待广大读者对本套规划教材提出宝贵意见，以便进一步修订，使该套规划教材不断完善。

21 世纪高等院校规划教材编委会

2004 年 8 月

前　言

数据结构是与数据相关的一门重要学科，不论是想通过升学考试还是想把程序编写得有水平，都要对这门学科下一点功夫才行。

笔者教授数据结构已有多年了（大概大于 10 年吧！），对于其内容已了如指掌，知道学生在研究上有哪些盲点，因此在内容的规划上尽可能让读者有事半功倍的学习效果。

本书的内容按不同的主题分为 14 章，每一章的每一小节均附有练习题及类似题，旨在让读者测试对该小节所谈及的内容是否已全部了解。在每一章的最后有动动脑时间，每一题的前面均加上此题的相关章节，如[1.2]表示此题目与 1.2 节相关，读者可以参考。

在每一章重要的主题后均附有程序加以测试，期使读者对理论能进一步认识与了解，尤其是第四章的链接表，笔者用一种更有效率的算法（algorithms）进行执行，希望读者能与笔者分享这份甜美果实。

本书的繁转简工作由董国平、林晓珊、李强、黄浩、林丽、王晓青、李欣、杨勇、林广毅、刘涛、曲波等人完成，在此对他们所做的工作表示感谢，也感谢中国水利水电出版社的编辑们，是他们的认真工作使得本书得以尽快地与读者见面。

最后，感谢邱义伦在程序上的测试，使得本书能够早日完成。笔者才疏学浅，若有叙述不详之处，盼各位批评与指教。

编　者

2005 年 10 月

目　　录

第 1 章　算法分析

算法是解决某一问题的有限步骤，而评判算法的优劣可利用 Big-O（复杂度）进行分析，如 O(n)比 O(n^2)效率高。本章旨在介绍如何计算 Big-O，从而知道一个算法是否优于另外一个算法。

1.1　算法

算法（Algorithms）是解决问题（problems）的有限步骤程序。举例来说，有下面这个问题：判断字符 x 是否存在于一个已排序好的字符串 s 中，其算法为从 s 串的第一个元素开始，依次进行比较，直到发现字符 x 或者到达 s 串尾部，假如字符 x 被找到，则打印出 Yes，否则打印出 No。

可是当问题很复杂时，上面描述的算法就难以表达出来。因此，算法大都用类似的程序语言来描述，从而利用读者所熟悉的程序语言执行。本书直接用 C 程序语言来描述，因此读者应具备编写 C 程序语言的能力。

读者是否常常会问这样的一个问题："这个人写的程序比另外一个人写的程序好吗？"，答案不是因为这个人是班上第一名，因此所写出的程序一定就是最好的，而是应该利用客观的方法进行比较，而此客观的方法就是复杂度分析（complexity analysis）。首先必须求出程序中每一基本操作的执行次数（其中{和}不加以计算），将这些执行次数加起来，然后求出其 Big-O。看一下以下 6 个范例。

1.1.1　数组元素相加（Add array members）

将数组中每个元素相加后返回总和。实现该算法的程序段如下所示：

```
                                执行次数
int sum(int arr[], int n)
{
int i, total=0;                 1
for(i=0; i<n; i++)              n+1
      total+=arr[i];            n
return total;                   1
}
```

上述程序段中语句的执行次数总和为 1+n+1+n+1=2n+3。

1.1.2　矩阵相乘（Matrix Multiplication）

矩阵相乘的定义如下：

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \times \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} = \begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ z_1 & z_2 & z_3 \end{bmatrix}$$

其中 $x_1=a_1*1+a_2*4+a_3*7$

$x_2=a_1*2+a_2*5+a_3*8$

$x_3=a_1*3+a_2*6+a_3*9$，其余依此类推。

实现该算法的程序段如下所示：

```
void mul(int a[][], int b[][], int c[][], int n)                    执行次数
{
int i,j,k,sum;
for(i=0;i<n; i++)                                                   1
     for (j=0;j<n;j++)    {                                         n+1
          sum=0;                                                    n(n+1)
          for (k=0;k<n;k++)                                         n^2
               sum=sum+a[i][k] * b[k][j];                           n^2(n+1)
          c[i][j] = sum;                                            n^3
     }                                                              n^2
}
```

上述程序段中语句的执行次数总和为 $2n^3+4n^2+2n+2$。

1.1.3 顺序查找（Sequential search）

顺序查找表示在一数组中，由第 1 个元素开始依次查找，直到找到欲查找的数据或到达数组最后一个元素。

实现该算法的程序段如下所示：

```
int search(int data[],int target,int n)
{
int i;
for(i=0;i<n;i++)
     if(target == data[i])
          return i;
}
```

上述程序中语句的执行次数总和将在后面给出。

1.1.4 折半查找（Binary search）

折半查找不同于顺序查找，其具体描述请见后面所述，其效率较佳。

实现该算法的程序段如下所示：

```
int search(int data[], int target, int n)
{
int i, mid, lower=0, upper=n-1;
mid=(lower+upper)/2;
while(lower<upper) {
     if(data[mid]==target)
          return mid;
     else
          if(data[mid] > target)
               upper = mid - 1;
```

```
        else
            lower = mid + 1;
        mid = (lower + upper)/2;
    }
}
```

上述程序中语句的执行次数总和将在后面给出。

1.1.5　斐波那契（Fibonacci）数列（递归的程序段）

斐波那契数列表示第 n 项为第 n-1 项和第 n-2 项的和，如 0，1，1，2，3，5，8，13…是一个斐波那契数列。

使用递归方式实现该算法的程序段如下所示：

```
int Fibonacci(int n)
{
if(n == 0)
    return 0;
else
    if (n == 1)
        return 1;
    else
        return (Fibonacci(n-1) + Fibonacci(n-2));
}
```

上述程序中语句的执行次数总和将在后面给出。

1.1.6　斐波那契数列（非递归的程序片段）

使用非递归方式实现该算法的程序段如下所示：

```
int Fibonacci(int n)
{
int prev1,prev2,item, i;
if (n == 0)
    return 0;
else
    if (n == 1)
        return 1;
    else  {
            prev2 = 0;
            prev1 = 1;
            for ( i = 2;i<=n;i++)   {
                item = prev1 + prev2;
                prev2 = prev1;
                prev1 = item;
            }
            return item;
        }
}
```

上述程序中语句的执行次数总和将在后面给出。

练 习 题

1．计算出下列程序段中 x=x+1 语句的执行次数。

```
(1) int i;
for(i=1;i<=100;i+=2)
x=x+1;
(2) int i=1;
while(++i <=100)
x=x+1;
(3) int i=1;
do
{
x=x+1;
} while(x++ <=100);
```

类 似 题

1．计算出下列程序段中 x=x+1 语句的执行次数。

```
(1) int i;
for(i=0;i<=100;i+=5)
x=x+1;
(2) int i=0;
while(i++<=100)
x=x+1;
```

1.2 Big-O（复杂度）

如何计算算法所需要的执行时间呢？在程序或算法中，每一条语句（statement）的执行时间包括以下两个部分，两个部分相乘即为此语句的执行时间。

- 此语句执行的次数。
- 每一次执行所需的时间。

由于每一语句所需的时间必须考虑到机器和编译器的功能，因此通常只考虑执行的次数。例如有下列三段程序，请计算出语句 x=x+1 的执行次数：

```
  ⋮                for(i=1; i<=n; i++)          for(i=1; i<=n; i++)
  ⋮                  {  x=x+1;                    for(j=1; j<=n; j++)
x=x+1;                    ⋮                              x=x+1;
  ⋮                  }                                    ⋮
 (a)                    (b)                          (c)
```

（a）的执行次数为 1 次，（b）的执行次数为 n 次，（c）的执行次数为 n^2 次。

在算法分析时，一般称语句的执行次数为 order of magnitude，所以上述（a）、（b）、（c）

中，语句 x=x+1 的 order of magnitude 分别为 1、n、n^2，而整个算法的 order of magnitude 为算法中每一条语句的执行次数之和。

算完程序语句的执行次数后，通常利用 Big-O（复杂度）来表示此算法执行的时间。

如果有两个常数 c 与 n_0，当所有 $n>n_0$ 时都满足 $f(n)<cg(n)$，则 $f(n)=O(g(n))$。

上述定义表示可以找到 c 和 n_0，使得 $f(n)<cg(n)$，此时，可以说 f(n)的 Big-O 为 g(n)。请看下列范例：

（a）$3n+2=O(n)$，因为可以找到 $c=4$，$n_0=2$，使得 $3n+2<4n$。

（b）$10n^2+5n+1=O(n^{2)}$，因为可以找到 $c=11$，$n_0=6$ 使得 $10n^2+5n+1<11n^2$。

（c）$7*2^n+n^2+n=O(2^n)$，因为可以找到 $c=8$，$n_0=4$ 使得 $7*2^n+n^2+n<8*2^n$。

（d）$10n^2+5n+1=O(n^3)$，这可以很清楚地看出，原来 $10n^2+5n+1\in O(n^2)$，而 n^3 又大于 n^2，理所当然 $10n^2+5n+1=O(n^3)$是没有问题的。同理也可以得知 $10n^2+5n+1\neq O(n)$，因为 f(x)没有小于等于 cg(n)。

另外，数组元素的 Big-O 为 O(n)，起泡排序为 $O(n^2)$，而矩阵乘法为 $O(n^3)$，顺序查找为 O(n)，折半查找为 O(logn)，而斐波那契数列若以递归处理则为 $O(2^{n/2})$，若以非递归方式处理则为 O(n)。

其实，可以对下列公式加以证明，

当 $f(n)=a_mn^m+\cdots+a_1n+a_0$ 时，$f(n)=O(n^m)$

证 明

$$f(n)\le\sum_{i=0}^{m}|a_i|n^i$$

$$\le n^m.\sum_{i=0}^{m}|a_i|n^{i-m}$$

$$\le n^m.\sum_{i=0}^{m}|a_i|\quad (n>1)$$

所以 $f(n)=O(n^m)$。

即 Big-O 取其最大指数的部分即可，因此前面叙述的范例中，数组元素相加的 Big-O 为 O(n)，起泡排序为 $O(n^2)$，而矩阵相乘为 $O(n^3)$。

顺序查找（sequential search）的情形可分为 3 种：第一种为最坏的情形，即要查找的给定值是表中的最后一个，因此需要 n 次才能查找到给定值（假设表中含有 n 个值）；第二种为最好的情形，此情形与第一种刚好相反，表示欲查找的给定值在第一个，故只要 1 次便可查找到给定值；最后一种为平均情况，其平均查找到的次数如下面公式所示：

$$\sum_{k=1}\left(k\times\frac{1}{n}\right)=\frac{1}{n}\times\sum_{k=1}^{n}k=\frac{1}{n}(1+2+...n)=\frac{1}{n}\times\frac{n(n+1)}{2}=\frac{n+1}{2}$$

因此顺序查找的 Big-O 为 O(n)。

折半查找（binary search）的情形和顺序查找不同，折半查找法是表已经排序好，因此由中间（mid）开始比较，便可知欲查找的给定值落在 mid 的左边还是右边，再将左边的中间拿

出来与关键字（key）相比，只是每次要调整每个部分的起始位置或最终位置，如图 1-1 所示。

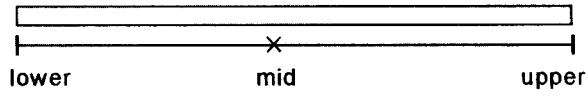

图 1-1 折半查找初始状态图

当 key>data[mid]时，mid=(lower+upper)/2，则 lower=mid+1，upper 不变，如图 1-2 所示。

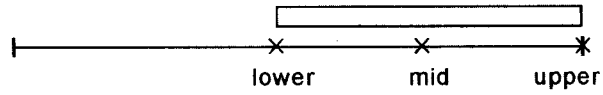

图 1-2 当 key>data[mid]时的折半查找状态图

当 key<data[mid]时，则 upper=mid-1，lower 不变，如图 1-3 所示。

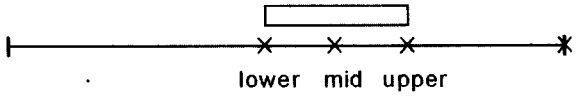

图 1-3 当 key<data[mid]时的折半查找状态图

若此时 key=data[mid]，便找到了欲查找的给定值，从上可知当 data 数组的大小为 32 时，其查找的点如图 1-4 所示，假设 key 大于 data 数组中所有元素值。

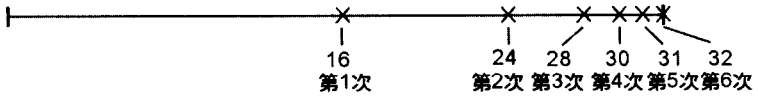

图 1-4 当 key=data[mid]时的折半查找状态图

查找的次数为 6=log32+1，此处的 log 表示 $\log_2$，数据量为 128 个时，其查找的次数为 log128+1，因此当表中含有 n 条记录时，其执行的次数为 logn+1。表 1-1 为折半查找与顺序查找的对比表，假设 key 大于数组中所有元素值。

表 1-1 折半查找与顺序查找的对比表

数组大小	折半查找	顺序查找
128	8	128
1,024	11	1024
1,048,576	21	1,048,576
4,294,967,296	33	4,294,967,296

从表 1-1 中，读者大略可知折半查找比顺序查找效率高，其执行效率为 O(logn)。

接下来讨论一个更有趣的问题——斐波那契数列（Fibonacci number），其定义如下：

$f_0=0$

$f_1=1$

$f_n=f_{n-1}+f_{n-2}$，$n\geqslant 2$