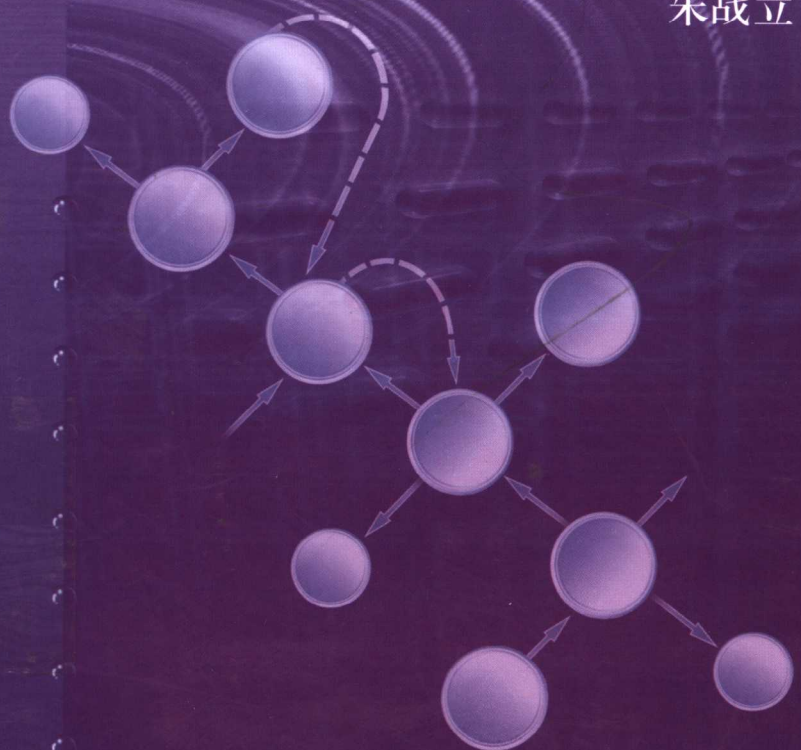
数据结构

——使用C语言

（第3版）

典型题解与上机实验指导

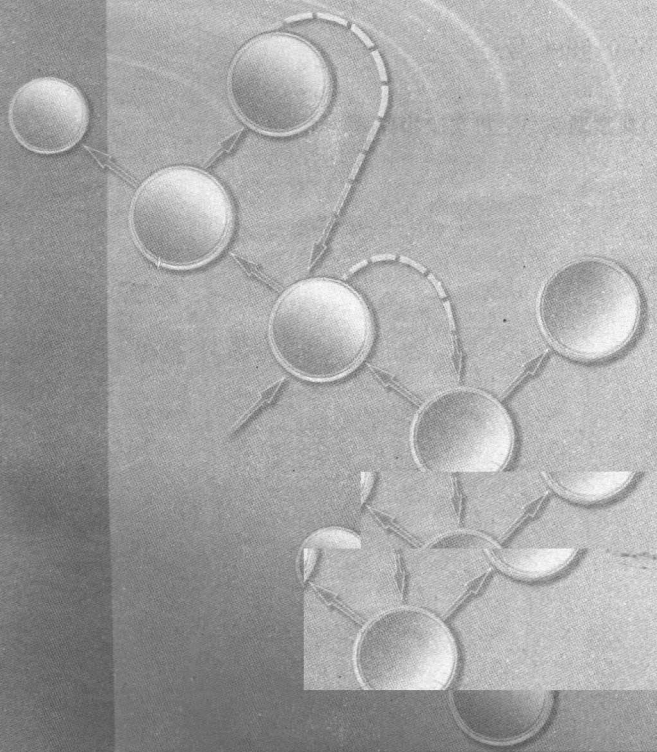
朱战立　张选平　韩家新　编



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS

数据结构
——使用C语言
（第3版）
典型题解与上机实验指导

朱战立　张选平　韩家新　编



西安交通大学出版社
XI'AN JIAOTONG UNIVERSITY PRESS
·西安·

内容简介

本书是《数据结构——使用C语言》(第3版)的配套学习参考书。

本书共分11章,前10章包括了线性表、堆栈、队列、串、数组、递归算法、树、二叉树、图、排序、查找等典型数据结构课程内容。每章主要由基本内容和典型题解两部分组成。基本内容部分简述了该章的学习要求,以及基本术语和基本概念,是该章学习和考试复习的大纲;典型题解部分的例题主要分为概念题和算法设计题,都是作者精心选编的,具有典型性。另外,大部分章节最后都包含了一个上机实习典型题解的例子,第11章搜集整理了几十道上机实验题目,并分别按节整理列出。这些内容可以帮助教师和学生完成上机实习和课程设计的教学过程。

本书可作为计算机本科和专科学生、报考计算机专业硕士研究生考生的学习参考书。

图书在版编目(CIP)数据

数据结构——使用C语言(第3版)典型题解与上机实验指导/朱战立,张选平,韩家新编.—西安:西安交通大学出版社,2007.2
ISBN 978-7-5605-2416-0

Ⅰ.数… Ⅱ.①朱…②张…③韩… Ⅲ.C语言-程序设计 Ⅳ.TP312

中国版本图书馆CIP数据核字(2007)第016996号

书　　名:数据结构——使用C语言(第3版)典型题解与上机实验指导
编　　者:朱战立　张选平　韩家新
出版发行:西安交通大学出版社
地　　址:西安市兴庆南路25号(邮编:710049)
电　　话:(029)82668357　82667874(发行部)
　　(029)82668315　82669096(总编办)
印　　刷:西安交通大学印刷厂
字　　数:324千字
开　　本:787 mm×1 092 mm　1/16
印　　张:13.5
版　　次:2007年2月第1版　2007年2月第1次印刷
书　　号:ISBN 978-7-5605-2416-0/TP·488
定　　价:18.00元

前　言

本书是作者编著出版、并经不断修改和完善、目前已出版到第3版的《数据结构——使用C语言》的配套学习参考书。《数据结构——使用C语言》一书自1997年出版以来，深受广大教师和学生欢迎，虽然教材经反复修改和完善，已经出版至第3版，但由于作者工作繁忙，一直没有能够出版配套的习题解辅导教材。经西安交通大学出版社领导和编辑一再鼓励，作者终于完成了目前这本配套辅导教材。

本书共分11章，前10章分别为线性表、堆栈、队列、数组、串、递归算法、广义表、树、二叉树、图、排序、查找等典型数据结构课程内容。每章主要由基本内容和典型题解两部分组成。为辅导学生完成上机实习和课程设计题，大部分章节中还包含了上机实习题解部分。基本内容部分简述了该章的学习要求，以及基本术语和基本概念，是该章学习和考试复习的大纲；典型题解部分的例题主要分为概念题和算法设计题，这些例题都是作者精心选编的，具有典型性。作者在解题时注意结合基本概念进行分析，对读者理解学习内容和掌握解题方法很有帮助。有些例题后边还有对类似问题的深入讨论。数据结构课程学习要和上机实习过程相结合，许多学生在完成上机实习题和课程设计题时感觉非常困难，上机实习典型题解部分正是专为解决这些学生的问题设计的。另外，第11章搜集整理了几十道上机实验题目，并分别按节整理列出，这既可以帮助教师安排上机实习内容和课程设计题目，也可以方便学生自己上机完成较复杂的算法设计问题。

本书的算法用C语言描述，且所有算法和上机实习题程序都在计算机上调试通过。

本书可作为计算机本科和专科学生、报考计算机专业硕士研究生考生的学习参考书。

本书第7章、第8章和第9章由朱战立和张选平共同编写，第10章和第11章由朱战立和韩家新共同编写，其余各章和全部附录由朱战立编写。全书由朱战立修改定稿。

尽管作者在写作过程中非常认真和努力，但由于水平有限，错误和不足之处在所难免，敬请读者批评指正。

编　者

2006.11

目　录

第1章 绪 论

1.1 基本内容

1.1.1 学习要求

本章基本知识点：

- ☞ 数据结构的基本概念
- ☞ 数据结构的基本术语
- ☞ 数据类型和抽象数据类型
- ☞ 算法和算法设计目标
- ☞ 算法效率度量

本章重点：

- ☞ 数据结构的基本概念
- ☞ 算法分析入门

本章难点：

- ☞ 基本概念和基本术语
- ☞ 算法分析入门

1.1.2 基本概念

数据结构：是一门研究非数值计算的程序设计问题中，计算机的操作对象以及它们之间关系和操作的课程。数据结构被形式地定义为(D, R)，其中D是数据元素的有限集合，R是D上的关系。

数据：人们利用文字符号、数字符号以及其他规定的符号，对现实世界的事物及其活动所作的抽象描述。

数据元素：表示数据的基本单位。数据元素通常由若干个数据项组成。

数据的逻辑结构：数据元素之间的联系结构。

数据的存储结构：数据的逻辑结构在计算机中的存储方式，也称数据的物理结构。数据存储结构的基本形式有两种：一种是顺序存储结构，另一种是链式存储结构。

顺序存储结构：把数据元素存储在一块连续地址空间的内存中，其特点是逻辑上相邻的数据元素在物理上也相邻，数据间的逻辑关系表现在数据元素的存储位置关系上。

链式存储结构：使用指针把发生联系的数据元素链接起来，其特点是逻辑上相邻的数据元

素在物理上不一定相邻,数据间的逻辑关系表现在结点的链接关系上。

数据的操作:允许对数据元素进行的操作(或称运算)。

类型:一组值的集合。例如,整数类型通常是-32768～32767。

数据类型:指一个类型和定义在这个类型上的操作集合。例如,int类型是集合[-32768,32767]和定义在这个集合上的加(+)、减(-)、乘(*)、除(\)及求模(%)运算。

抽象数据类型:指一个逻辑概念上的类型和这个类型上的操作集合。例如,圆的抽象数据类型包括描述圆的基本数据(圆的半径)和常规的运算(如求圆的面积、体积等)。

算法:对特定问题求解步骤进行描述的计算机指令的有限序列。

算法需满足的性质:输入性、输出性、有限性、确定性、可执行性。

算法的时间复杂度:算法执行时间和数据元素个数 n 之间的函数关系,也称作算法的时间效率。通常采用 $O(f(n))$ 表示法。

算法的空间复杂度:执行算法所需内存空间和数据元素个数 n 之间的函数关系,也称作算法的空间效率。通常采用 $O(f(n))$ 表示法。

1.2　典型题解

1.2.1　概念题

例1-1　填空题

(1)数据结构是一门研究非数值计算的程序设计问题中,计算机的(　)以及它们之间(　)和操作的课程。

(2)数据结构被形式地定义为 (D, R),其中 D 是(　)的有限集合,R 是 D 上的(　)。

(3)数据结构包括数据的(　)、数据的(　)和数据的(　)这三个方面的内容。

(4)数据结构按逻辑结构可分为两大类,它们分别是(　)和(　)。

(5)数据的操作最常用的有五种,分别是(　)、(　)、(　)、(　)和(　)。

(6)算法的效率可分为(　)效率和(　)效率。

答:(1)操作对象;关系

(2)数据元素;关系

(3)逻辑结构;存储结构;操作

(4)线性结构;非线性结构

(5)插入、删除、修改、查找、排序

(6)时间;空间

例1-2　选择题

(1)非线性结构是数据元素之间存在一种_____。

A. 一对多关系　　B. 多对多关系

C. 多对一关系　　D. 一对一关系

(2)数据结构中,与所使用的计算机无关的是数据的_____。

A. 存储结构　　B. 物理结构

C. 逻辑结构　　D. 物理和存储结构

(3)算法分析的目的是_____。

A. 找出数据结构的合理性　　B. 研究算法中的输入和输出的关系

C. 分析算法的效率以求改进　　D. 分析算法的易懂性和文档性

(4)算法分析的两个主要方面是_____。

A. 时间复杂性和空间复杂性　　B. 正确性和简明性

C. 可读性和文档性　　D. 数据复杂性和程序复杂性

(5)算法指的是_____。

A. 计算方法　　B. 排序方法

C. 解决问题的有限运算序列　　D. 调度方法

(6)计算机算法必须具备输入、输出和_____等五个特性。

A. 可行性、可移植性和可扩充性　　B. 有限性、确定性、可执行性

C. 确定性、有穷性和稳定性　　D. 易读性、稳定性和安全性

答:(1) B　(2) C　(3) C　(4) A　(5) C　(6) B

例 1-3　简答题

(1)数据结构和数据类型两个概念之间有区别吗?

(2)简述线性结构与非线性结构的不同点。

答:(1)简单地说,数据结构定义了一组按某些关系结合在一起的数组元素。数据类型不仅定义了一组带结构的数据元素,而且还在其上定义了一组操作。

(2)线性结构反映数据元素间的逻辑关系是一对一的,非线性结构反映数据元素间的逻辑关系是多对多的。

例 1-4　设三个函数 $f()$,$g()$和 $h()$分别为:

$$f(n)=100n^3+n^2+1000$$

$$g(n)=25n^3+5000n^2$$

$$h(n)=n^{1.5}+5000n\log_2 n$$

请判断下列关系是否成立:

(1)$f(n)=O(g(n))$

(2)$g(n)=O(f(n))$

(3)$h(n)=O(n1.5)$

(4)$h(n)=O(n\lg n)$

答:数学符号"O"的严格的数学定义:若 $T(n)$和 $f(n)$是定义在正整数集合上的两个函数,则 $T(n)=O(f(n))$表示存在正的常数 C 和 n_0,使得当 $n\leqslant n_0$时,满足 $0\leqslant T(n)\leqslant C\times f(n)$。

通俗地说,就是当 $n\to\infty$时,$f(n)$的函数值增长速度与 $T(n)$的增长速度同阶。一般情况下,一个函数的增长速度与该函数的最高次阶同阶。

因此,$O(f(n))=n^3$,　$O(g(n))=n^3$,　　$O(h(n))=n^{1.5}$

所以有:(1)成立;(2)成立;(3)成立;(4)不成立。

例 1-5　设有两个算法在同一机器上运行,其执行时间分别为 $100n^2$ 和 2^n,要使前者快于后者,n 至少要多大?

答:要使前者快于后者,即前者的时间消耗低于后者,即 $100n^2<2^n$。求解,可得 $n=15$。

例 1-6　按增长率由小至大的顺序排列下列各函数:

2^{100}, $(3/2)^n$, $(2/3)^n$, n^n, $n^{0.5}$, $n!$, 2^n, $\lg n$, $n^{\lg n}$, $n^{(3/2)}$

答:常见的时间复杂度按数量级递增排列,依次为:常数阶 $O(1)$、对数阶 $O(\lg n)$、线性阶 $O(n)$、线性对数阶 $O(n\lg n)$、平方阶 $O(n^2)$、立方阶 $O(n^3)$、k 次方阶 $O(n^k)$、指数阶 $O(2^n)$。

先将题中的函数分成如下几类:

常数阶:2^{100}

对数阶:$\lg n$

k 次方阶:$n^{0.5}$,$n^{3/2}$

指数阶(按指数由小到大排):$n^{\lg n}$,$(3/2)^n$,2^n,$n!$,n^n

注意:$(2/3)^n$ 由于底数小于1,所以是一个递减函数,其数量级应小于常数阶。

根据以上分析按增长率由小至大的顺序可排列如下:

$(2/3)^n < 2^{100} < \lg n < n^{0.5} < n^{(3/2)} < n^{\lg n} < (3/2)^n < 2^n < n! < n^n$

1.2.2　算法分析题

例1-7　设数组a和b在前边部分已赋值,求如下两个 n 阶矩阵相乘运算程序段的时间复杂度。

```
for(i = 0; i < n; i++)
        for(j = 0; j < n; j++)
        {
            c[i][j] = 0;                                    //基本语句1
            for(k = 0; k < n; k++)
                c[i][j] = c[i][j] + a[i][k] * b[k][j];      //基本语句2
        }
```

答:设基本语句的执行次数为 $f(n)$,有:

$$f(n) = n2 + n3$$

因程序段的时间复杂度 $T(n) = f(n) = n2 + n3 \leqslant c \times n3 = O(n3)$,其中 c 为常数,所以该程序段的时间复杂度为 $O(n3)$。

例1-8　设 n 为如下程序段处理的数据个数,求如下程序段的时间复杂度。

```
for(i = 1; i <= n; i = 2 * i)
    printf("i = %d\n", i);          //基本语句
```

答:设基本语句的执行次数为 $f(n)$,有 $2f(n) \leqslant n$,即有 $f(n) \leqslant lbn$。

因程序段的时间复杂度 $T(n) = f(n) \leqslant lbn \leqslant c \times lbn = O(lbn)$,其中 c 为常数,所以该程序段的时间复杂度为 $O(lbn)$。

在很多情况中,算法中数据元素的取值情况、不同算法的时间复杂度也会不同。此时算法的时间复杂度应是数据元素最坏情况下取值的时间复杂度,或数据元素等概率取值情况下的平均(或称期望)时间复杂度。

例1-9　下边的算法是用冒泡排序法对数组a中的 n 个整数类型的数据元素(a[0]~a[n-1])从小到大排序的算法,求该算法的时间复杂度。

```
void BubbleSort(int a[], int n)
{
    int i, j, flag = 1;
```

```
    int temp;

    for(i = 1; i < n && flag = = 1; i + + )
    {
        flag = 0;
        for(j = 0; j < n - i; j + + )
        {
            if(a[j] > a[j + 1])
            {
                flag = 1;
                temp = a[j];
                a[j] = a[j + 1];
                a[j + 1] = temp;
            }
        }
    }
}
```

答:这个算法的时间复杂度随待排序数据的不同而不同。当某次排序过程中没有任何两个数组元素交换位置,则表明数组元素已排序完毕,此时算法将因标记 flag=0 不满足循环条件而结束。但是,在最坏情况下,每次排序过程中都至少有两个数组元素交换位置,因此,应按最坏情况计算该算法的时间复杂度。

设基本语句的执行次数为 $f(n)$,最坏情况下有:

$$f(n) \approx n + 4 \times n2/2$$

因算法的时间复杂度 $T(n) = f(n) \approx n + 2 \times n2 \leqslant c \times n2 = O(n2)$,其中 c 为常数,所以该算法的时间复杂度为 $O(n2)$。

例 1-10　下边算法是在一个有 n 个数据元素的数组 a 中,删除第 i 个位置的数组元素,要求当删除成功时数组元素个数减 1,求该算法的时间复杂度。其中,数组下标从 0 至 $n-1$。

```
int Delete(int a[], int * n, int i)
{
    int j;
    if(i < 0 || i > * n) return 0;                    //删除位置错误,失败返回
    for(j = i + 1; j < * n; j + + ) a[j - 1] = a[j];
                                                      //顺次移位填补
    ( * n) - - ;                                      //数组元素个数减 1
    return 1;                                         //删除成功返回
}
```

答:这个算法的时间复杂度随删除数据的位置不同而不同。当删除最后一个位置的数组元素时有 $i = n-1$, $j = i+1 = n$,此时因不需移位填补,而循环次数为 0;当删除倒数最后一个

位置的数组元素时有 $i=n-2$，$j=i+1=n-1$，此时因只需移位填补一次而循环次数为1；依此类推，当删除第一个位置的数组元素时有 $i=0$，$j=i+1=1$，此时因需移位填补 $n-1$ 次，而循环次数为 $n-1$。此时算法的时间复杂度应是删除数据的位置等概率取值情况下的平均时间复杂度。

假设删除任何位置上的数据元素都是等概率的(一般情况下均可作等概率假设)，设 Pi 为删除第 i 个位置上数据元素的概率，则有 $Pi=1/n$，设 E 为删除数组元素的平均次数，则有：

$$E=\frac{1}{n}\sum_{i=0}^{n-1}(n-1-i)=\frac{1}{n}[(n-1)+(n-2)+\cdots+2+1+0]$$

$$=\frac{1}{n}\cdot\frac{n(n-1)}{2}=\frac{n-1}{2}$$

因该算法的时间复杂度 $T(n)=E\leqslant(n+1)/2\leqslant c\times n=O(n)$，其中 c 为常数，所以该算法的时间复杂度为 $O(n)$。

1.2.3　算法设计题

例1-11　设计一个从2个整数类型数据中得到较大数值的算法。

算法设计如下：

```
int Max1(int x1, int x2)
{
    if(x1 >= x2) return x1;
    else return x2;
}
```

说明：在上述算法的设计中，函数名具有整数类型的返回值，用于返回得到的较大数值；两个参数均为输入参数(即值参)，参数的类型均为整数类型。

例1-12　设计一个从3个整数类型数据中得到最大数值和次大数值的算法。

设计分析如下：

算法要有3个输入参数，分别表示3个整数类型数据；由于算法要带回2个返回值，而函数名只能带回1个返回值，因此必须设计两个输出参数，用于带回算法的2个结果值，其类型为整数的指针类型。

算法设计如下：

```
void Max2(int x1, int x2, int x3, int *y1, int *y2)
//输出参数 y1 和 y2 设计为指针类型
{
    if(x1 >= x2 && x1 >= x3)
    {
        *y1 = x1;
        if(x2 >= x3) *y2 = x2;
        else *y2 = x3;
    }
```

```
    if(x1 >= x2 && x1 < x3)
    {
        *y1 = x3;
        if(x1 >= x2) *y2 = x1;
        else *y2 = x2;
    }

    if(x1 < x2 && x2 >= x3)
    {
        *y1 = x2;
        if(x1 >= x3) *y2 = x1;
        else *y2 = x3;
    }

    if(x1 < x2 && x2 < x3)
    {
        *y1 = x3;
        if(x1 >= x2) *y2 = x1;
        else *y2 = x2;
    }
}
```

第 2 章　线性表

2.1　基本内容

2.1.1　学习要求

本章基本知识点：

- ☞ 线性表的逻辑结构及其基本操作
- ☞ 顺序存储结构和链式存储结构
- ☞ 线性表的顺序存储结构，以及顺序存储结构线性表的基本操作的算法设计
- ☞ 线性表的链式存储结构，以及链式存储结构线性表的基本操作的算法设计
- ☞ 静态链表的存储结构
- ☞ 线性表的应用和具体设计方法

本章重点：

- ☞ 线性表的逻辑结构
- ☞ 顺序存储结构和链式存储结构
- ☞ 线性表的两种存储结构及其操作的算法设计

本章难点：

- ☞ 线性表的两种存储结构及其操作的算法设计

2.1.2　基本概念

顺序存储结构：使用一片地址连续的、有限内存单元空间存储数据元素的一种计算机存储数据方法。

链式存储结构：把数据元素和指针定义成一个存储体，使用指针把发生联系的数据元素链接起来的一种计算机存储数据方法。

线性表：允许在任意位置进行插入或删除操作的、有 n 个数据元素的序列。

顺序表：用顺序存储结构存储的线性表。

单链表：用链式存储结构存储的、每个结点只有一个后继指针的线性表。

单循环链表：最后一个结点的指针指向头结点（或第一个数据元素结点）的单链表。

双向链表：用链式存储结构存储的、每个结点包括后继指针域和前趋指针域的线性表。

2.1.3　基本结构体定义

顺序表的结构体定义：

```
typedef struct
{
    DataType list[MaxSize];
    int size;
} SeqList;
```

说明:DataType 可指定为任意数据类型。例如,如果程序中有如下语句:

```
typedef int DataType;
```

就表示定义 DataType 为 int 数据类型。

单链表的结点结构体定义:

```
typedef struct Node
{
    DataType data;
    struct Node * next;
} SLNode;
```

2.2　典型题解

2.2.1　概念题

例 2-1　填空题

(1)在顺序表中插入或删除一个数据元素,需要平均移动(　)数据元素,具体移动的数据元素个数与(　)有关。

(2)向一个长度为 n 的顺序表的第 i 个数据元素($1 \leqslant i \leqslant n+1$)之前插入一个数据元素时,需向后移动(　)个数据元素。

(3)删除一个长度为 n 的顺序表的第 i 个数据元素($1 \leqslant i \leqslant n$)时,需向前移动个数据元素。

(4)顺序表中逻辑上相邻的数据元素,其物理位置(　)。单链表中逻辑上相邻的数据元素,其物理位置(　)。

(5)在单链表中,除了首元结点(或称第一个数据元素结点)外,其他结点的存储位置由(　)指示。

(6)在有 n 个结点的单链表中,要删除已知结点 $*p$,需找到它的(　),其时间复杂度为(　)。

答:(1)表中一半;表长和该数据元素在表中的位置

(2)$n-i+1$

(3)$n-i$

(4)必定;不一定

(5)其直接前驱结点的链域值

(6)直接前驱结点的地址;$O(n)$

例 2-2　判断正误题(并作出简要说明)