高等院校实践类系列教材

数据结构

学习指导·实验指导·课程设计

陈媛 何波 蒋鹏 刘洁 编著

机械工业出版社
CHINA MACHINE PRESS

高等院校实践类系列教材

数据结构

学习指导·实验指导·课程设计

陈媛 何波 蒋鹏 刘洁 编著

机械工业出版社

本书是在作者多年讲授数据结构课程及指导学生实验的教学实践经验的基础上编写而成的。作者力图通过指导学生的实践和大量习题的解析，帮助学生深入学习、掌握并灵活运用数据结构知识。

全书分为10章。第1章至第9章为与“数据结构”教材对应的知识要点、例题解析、习题和实验指导，包括绪论、线性表、栈和队列、串、数组和广义表、树和二叉树、图、查找、排序等内容，可以帮助学生提纲契领地掌握知识重点、巩固所学内容；第10章为课程设计指导，根据数据结构课程的教学重点，给出15个课程设计题目，每个题目都有明确的要求。书后附录中包括了习题、实验和课程设计题目的参考答案及学生应提交的课程设计报告的格式及范文。

本书内容自成一体，可脱离数据结构教材单独使用，也可配合教材使用，起到衔接课堂教学与实验教学、课下辅导的作用。本书可作为高等院校计算机专业、信息专业或其他相关专业学生学习“数据结构”和其他程序类课程的参考教材，或研究生入学考试的辅导材料，也可作为广大参加自学考试的人员和软件工作者的参考资料。

图书在版编目（CIP）数据

数据结构学习指导·实验指导·课程设计/陈媛等编著. —北京：机械工业出版社，2008.1
（高等院校实践类系列教材）
ISBN 978-7-111-23199-8

Ⅰ.数… Ⅱ.陈… Ⅲ.数据结构－高等学校－教学参考资料
Ⅳ.TP311.12

中国版本图书馆CIP数据核字（2007）第206371号

机械工业出版社（北京市百万庄大街22号 邮政编码100037）
责任编辑：田淑华
责任印制：洪汉军
北京振兴源印务有限公司印刷厂印刷
2008年2月第1版·第1次印刷
184mm×260mm · 17.25印张·423千字
0001－4000册
标准书号：ISBN 978-7-111-23199-8
定价：27.00元

凡购本书，如有缺页，倒页，脱页，由本社发行部调换
销售服务热线电话：（010）68326294
购书热线电话：（010）88379639 88379641 88379643
编辑热线电话：（010）88379739
封面无防伪标均为盗版

前　言

“数据结构”课程是计算机及相关专业最重要的专业基础课程之一，是进行程序设计的理论和技术基础，同时对于计算机专业其他课程的学习也是十分重要的。本书是由长期在一线从事教学工作、有着丰富授课经验的教师编写而成的。

一、结构安排

全书共分10章，分别为绪论、线性表、栈和队列、串、数组与广义表、树和二叉树、图、查找、排序、课程设计。

前9章每章的内容编排为知识要点、例题解析、习题和实验指导4个部分。“知识要点”归纳基本内容，明确学习要求和重点；“例题解析”讨论解题思路、方法与技巧；“习题”涉及了本章的有关概念、方法和原理，分为基础题和综合题，基础题分单项选择题与填空题，习题中既收集了一些较容易的内容，也收集了有一定难度的研究生入学试题，既注意题目涉及的内容全面，又注意了题目难度的循序渐进；“实验指导”给出本章实验目的和实验内容，实验内容根据难度有填空题和算法编写题，以适应不同读者的需要。

第10章为课程设计指导，根据数据结构课程的教学重点，给出15个课程设计题目，每个题目都有明确的要求，题目内容尽量与现实结合，要求学生综合应用数据结构课程的所有知识点及算法，熟练解决实际问题，为学生将来编写大型软件打下良好的基础。

附录中的参考答案部分是习题、实验、课程设计的解答，为教师和学生提供了习题解答的参考方案。

本书由陈媛主编，并编写了第10章，负责全书的统编；何波编写了第6章、第7章；蒋鹏编写了第8章、第9章；刘洁编写了第1章至第5章；本书附录由各章节负责人分别编写。

二、本书特点

本书是在作者多年讲授数据结构课程及指导学生实验的教学实践经验的基础上，参考了近年来出版的多种数据结构类书籍和全国各高校历年来研究生入学考试试题编写而成的。本书给出的所有算法和程序采用C语言描述，并均调试通过。本书内容全面，层次分明，结构合理，知识点清晰，有利于加深对课程的理解，帮助学生从广度和深度上把握知识体系，拓宽解题思路，提高分析解决实际问题的编程能力。

三、适用对象

本书内容自成一体，可脱离“数据结构”教材单独使用，也可配合教材使用，起到衔接课堂教学与实验教学、课后辅导的作用。

本书可作为高等院校计算机专业、信息专业或其他相关专业学生学习“数据结构”和其他程序设计类课程的参考教材，或研究生入学考试的辅导材料，也可作为广大参加自学考试的人员和软件工作者的参考资料。

限于编者的水平，书中错误和不妥之处，敬请读者批评指正。

编著者

2007年8月

目　录

第1章 绪 论

1.1 本章内容

1.1.1 基本内容

数据、数据元素、数据对象、数据结构、存储结构和数据类型等概念术语的确定含义；抽象数据类型的定义、表示和实现方法；算法的定义、设计的基本要求以及从时间和空间角度分析算法的方法。

1.1.2 学习要点

(1) 熟悉各名词、术语的含义，掌握基本概念，特别是数据的逻辑结构和存储结构之间的关系。分清哪些是逻辑结构的性质，哪些是存储结构的性质。熟悉逻辑结构的4种基本类型和存储结构两种基本机内表示方法。

(2) 理解算法5个要素的确切含义：1)动态有穷性(能执行到结束)；2)确定性(对于相同的输入执行相同的路径)；3)有输入；4)有输出；5)可行性(用以描述算法的操作都是可实现的)。

(3) 掌握计算语句频度和估算算法时间复杂度的方法。

1.1.3 例题解析

例 1-1 有下列几种用二元组表示的数据结构，画出它们分别对应的逻辑图形表示，并指出它们分别属于何种结构。

(1) L=(D,R),其中

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

R={r}

r={<5,1>,<1,3>,<3,8>,<8,2>,<2,7>,<7,4>,<4,6>}

(2) T=(D,R),其中

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

R={r}

r={<1,2>,<1,3>,<1,4>,<2,5>,<2,6>,<3,7>,<3,8>}

(3) G=(D,R),其中

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

R={r}

r={<1,2>,<2,1>,<1,4>,<4,1>,<2,3>,<3,2>,<2,6>,<6,2>,<2,7>,<7,2>,<3,7>,<7,3>,<4,6>,<6,4>,<5,7>,<7,5>}

解：(1) 对应的图形如图1-1所示。

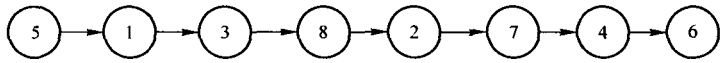

图 1-1　线性结构

在 L 中,每一个数据元素有且只有一个前驱元素(除第一个结点 5 外),有且只有一个后续(除最后一个元素 6 外),所以为线性结构。

(2) 对应的图形如图 1-2 所示。

在 T 中,每一个数据元素有且只有一个前驱元素(除根结点 1 外),但可以有任意多个后续结点(树叶可看成为具有 0 个后续结点),所以为树型结构。

(3) 对应的图形如图 1-3 所示。

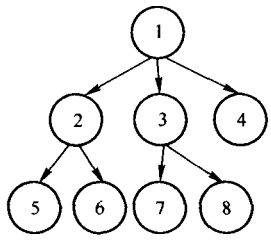

图 1-2　树型结构

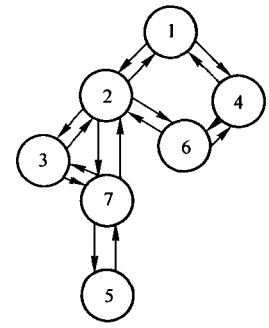

图 1-3　有向图型结构

在 G 中,每一个数据元素可以有任意多个前驱元素和任意多个后续元素,所以为有向图型结构。

例 1-2　求下列算法的时间复杂度。

(1)
```
int sum(int a[],int n)/* 累加求和 */
  {
   s=0;/* ①给累加变量 s 赋初值 */
   for (i =0;i <n;i ++) /* ②进行累加求和 */
   s+ =a[i];
   return(s);/* ③返回 s 的值 */
  }
```

(2)
```
matrixadd(int a[][n],int b[][n],int c[][n],int n) /* 矩阵相加 */
   /* a,b,c 分别为 n 阶矩阵,a,b 表示两个加数,c 表示和 */
  {
   for (i=0;i<n; ++i)
        for (j=0;j<n; ++j)
          c[i][j] = a[i][j]+ b[i][j];
  }
```

(3)
```
void bubble_ sort(int a[],int n)
  {
  /* 将 a 中整数序列重新排列成自小至大 */
```

```
/* 有序的整数序列。 */
   for (i=n-1,change=TRUE;i>1 && change; --i)
     {
      change = FALSE;
      for (j=0;j<i; ++j)
        if (a[j] > a[j+1]) { temp=a[j];a[j]=a[j+1];a[j+1]=temp;
                             change = TRUE;}
     }
} /* bubble_sort */
```

(4)
```
void select_sort(int a[],int n) {
/* 将 a 中整数序列重新排列成自小至大有序的整数序列 */
for ( i = 0;i< n-1; ++i )
{
 j = i;
 for ( k = i+1;k < n; ++k )
     if (a[k] < a[j] ) j = k;
 if ( j != i ) a[j]←→a[i]
} /* select_sort */
```

(5)
```
for (i=1;i<=n; ++i)
 for (j=1;j<=n; ++j)
    {
        c[i,j] = 0;
        for (k=1;k<=n; ++k)
            c[i,j] += a[i,k] * b[k,j];
}
```

解：

通常把算法中包含简单操作次数(也称为频度)的多少叫做算法的时间复杂性，它可看成是问题规模的函数，记为 T(n)。当算法较复杂时，只要大致计算出相应的数量级即可，即求算法的(渐近)时间复杂度。估算算法的(渐近)时间复杂度的常用方法是：

- 多数情况下，求最深层循环内的简单语句(原操作)的重复执行的次数。
- 当难以精确计算原操作的执行次数时，只需求出它关于 n 的增长率或阶即可。
- 当循环次数未知(与输入数据有关)，求最坏情况下的简单语句(原操作)的重复执行的次数。

(1) 第②步不是简单操作，可把它改写为：

```
i =0; /* 1 次 */
L1:if (i >=n) goto L2; /* n+1 次 */
 s+ =a[i]; /* n 次 */
 i ++ ;/* n 次 */
  goto L1/* n 次 */
 L2:return (s); /* ③ 1 次 */
```

因此该算法的时间复杂性为：$T(n)=4n+4=O(n)$。

(2) 通过与(1)相似的分析过程,可得到该算法的时间复杂性为:$T(n)=4n^2++5n+2=O(n^2)$。

(3) 基本操作为赋值操作,次数未知,最坏情况下的次数为 $n(n+1)/2$,时间复杂度:$O(n^2)$。

(4) 基本操作为比较(数据元素)操作,时间复杂度:$O(n^2)$。

(5) 基本操作为乘法操作,时间复杂度:$O(n^3)$。

1.2 习题

1.2.1 基础题

单项选择题

1. 数据对象是指_____。

A. 描述客观事物且由计算机处理的数值、字符等符号的总称

B. 数据的基本单位

C. 性质相同的数据元素的集合

D. 相互之间存在一种或多种特定关系的数据元素的集合

2. 在数据结构中,数据的基本单位是_____。

A. 数据项　　B. 数据类型　　C. 数据元素　　D. 数据变量

3. 数据结构中数据元素之间的逻辑关系被称为_____。

A. 数据的存储结构　B. 数据的基本操作　C. 程序的算法　　D. 数据的逻辑结构

4. 在数据结构中,与所使用计算机无关的是数据的_____。

A. 存储结构　　B. 逻辑和物理结构　C. 逻辑结构　　D. 物理结构

5. 在链式存储结构中,数据之间的关系是通过_____体现的。

A. 数据在内存的相对位置　　B. 指示数据元素的指针

C. 数据的存储地址　　D. 指针

6. 在定义 ADT 时,除数据对象和数据关系外,还需说明_____。

A. 数据元素　　B. 算法　　C. 基本操作　　D. 数据项

7. 计算算法的时间复杂度,属于一种_____。

A. 事前统计的方法　　B. 事前分析估算的方法

C. 事后统计的方法　　D. 事后分析估算的方法

8. 在对算法的时间复杂度进行估计的时候,下列最佳的时间复杂度是_____。

A. n^2　　B. nlogn　　C. n　　D. logn

9. 设使用某算法对 n 个元素进行处理,所需的时间是 $T(n)=100n\log_2 n+200n+2000$,则该算法的渐近时间复杂度为_____。

A. $O(1)$　　B. $O(n)$　　C. $O(200n)$　　D. $O(n\log_2 n)$

10. 有如下递归函数 fact(a),其时间复杂度为_____。

```
int fact(int a)
  {
    if(n==0)
```

```
    retrun 1;
   else
    return(n * fact(n-1));
}
```

A. $O(n)$　　B. $O(n^2)$　　C. $O(n^3)$　　D. $O(n^4)$

11. 线性表若采用链式存储结构时，要求内存中可用存储单元的地址_____。

A. 必须是连续的　　B. 部分地址必须是连续的

C. 一定是不连续的　　D. 连续不连续都可以

12. 线性结构的顺序存储结构是一种____①____的存储结构，线性表的链式存储结构是一种____②____的存储结构。

A. 随机存取　　B. 顺序存取　　C. 索引存取　　D. 散列存取

填空题

1. 数据结构由数据的____①____、____②____和____③____3部分组成。

2. 程序包括两个内容：____①____和____②____。

3. 数据结构在物理上可分为____①____存储结构和____②____存储结构。

4. 数据的物理结构，指数据元素在____①____中的表示，也即____②____。

5. 数据逻辑结构包括____①____、____②____和____③____3种类型，树型结构和有向图型结构合称为____④____。

6. 我们把每种数据结构均视为抽象类型，它不但定义了数据的____①____方式，还给出了处理数据的____②____。

7. 一个算法的时间复杂度是用该算法____①____的多少来度量的，一个算法的空间复杂度是用该算法在运行过程中所占用的____②____的大小来度量的。

8. 算法具有如下特点：____①____、可执行性、____②____、结果性、一般性。

9. 对于某一类特定的问题，算法给出了解决问题的一系列操作，每一操作都有它的____①____的意义，并在____②____内计算出结果。

10. 下面程序段的时间复杂度为_____。

```
i=1;
while(i<=n)
  i=i*3;
```

1.2.2 综合题

1. 简述数据结构的4种基本关系并画出它们的关系图。

2. 解释概念：(1)数据结构；(2) 抽象数据类型。

3. 算法的时间复杂度仅与问题的规模有关吗？为什么？

4. 在算法正确的情况下，应从哪几个方面来衡量一个算法的优劣性？

5. 求下面算法的功能及时间复杂度。

```
p=1.0;d=n;f=n;/*n为一正整数*/
while(d>0) {
```

```
    if(d%2==1) p=p*f;
    f=f*f;d=d/2;
}
```

6. 设 n 为正整数,试确定下列各程序段中前面有记号@的语句的频度。

(1)
```
i = 1;k = 0;
while (i<=n - 1) {
  @ k + = 10 * i;
    i ++;
  }
```

(2)
```
i =1;k=0;
do{
@ k+ =10* i;
    i ++;
}while (i <=n-1)
```

(3)
```
i = 1;k = 0;
  while (i<=n - 1) {
      i ++;
    @ k + = 10 * i;
          }
```

(4)
```
k = 0;
  for ( i = 1;i<=n ;i++ ) {
      for ( j = i;j<=n ;j++ )
              @ k ++;
   }
```

(5)
```
i=1;j=0;
    while (i+j<= n){
                  @ if (i>j) j++;
                  else i++;
                }
```

(6)
```
for( i=1;i<=n;i++ )
  for (j=1;j<=i;j++ )
    for (k=1;k<=j;k++ )
    @   x++;
```

(7)
```
x= n;y= 0;
  while (x>=(y+1) * (y + 1)){
        @ y ++;
      }
```

(8)
```
x= 91;y= 100;
 while (y >0){
      @ if (x>100){x- =10;y --;}
      else x ++ ;
}
```

7. 阅读下列算法：

```
void suan _ fa(int n)
{int i,j,k,s,x;
for(s=0,i=0;i<n;i++ )
   for(j=i;j<n;j++)
     s++;
i=1;j=n;x=0;
while(i<j)
{i++;j--;x+=2;}
printf("s=%d","x=%d",s,x);
}
```

(1) 分析算法中语句“s++;”的执行次数；

(2) 分析算法中语句“x+=2;”的执行次数；

(3) 分析算法的时间复杂度；

(4) 假定 n=5，试指出执行该算法的输出结果。

8. 求两个 n 阶矩形的乘法 C=A×B，其算法如下，分析该算法的时间复杂度。

```
#define MAX 100
void maxtrixmult(int n,float a[MAX][MAX],float b[MAX][MAX],float c[MAX][MAX])
{  int i,j,k;
   float x;
   for (i=1;i<=n;i++)                                    ①
   {
     for (j=1;j<=n;j++)                                  ②
     {  x=0;                                             ③
        for(k=1;k<=n;k++)                                ④
            x+=a[i][k]*b[k][j];                          ⑤
            c[i][j]=x;                                   ⑥
     }
   }
}
```

9. 试写一算法，自大至小依次输出顺序读入的 3 个整数 X，Y 和 Z 的值。

10. 试编写算法，计算 i! $*2^i$ 的值并存入数组 a[0…arrsize-1] 的第 i-1 个分量中(i= 1, 2,…,n)。假设计算机中允许的整数最大值为 maxint，则当 n>arrsize 或对某个 k($1\leqslant k\leqslant n$) 使 k! $*2^k$>maxint 时，应按出错处理，注意选择你认为较好的出错处理方法。

1.3 实验

1.3.1 验证算法的源程序结构及举例

目前数据结构教材上的算法一般用类 C 语言来描述，但要验证算法只能用标准 C 来验

证,所以存在类 C 语言与标准 C 的转换及验证算法的源程序编写问题。

1. 类 C 语言与标准 C 的转换要点

(1) 预定义常量和类型的问题。

在类 C 语言的算法中出现了下列常量,验证时就需在源程序中定义:

```
#define TRUE 1
#define FALSE 0
#define OK 1
#define ERROR 0
#define INFEASIBLE -1
#define OVERFLOW -2
#define int Status
```

(2) 算法描述的变量问题。

在类 C 语言算法(函数)中,辅助变量没有作变量说明,但在验证源程序的函数定义时,需对所有使用的变量(除函数参数外)作说明。

例 1-3 顺序表的插入算法中的变量说明

类 C 语言算法描述:

```
Status ListInsert_Sq(SqList &L,int pos,ElemType e)
{
  /* 在顺序线性表 L 的第 pos 个元素之前插入新的元素 e,pos 为位序 */
  /* pos 的合法值为 1≤pos≤Listlength_Sq(L)+1 */
  if (pos < 1 || pos > L.length+1) return ERROR;
  /* 插入位置不合法 */
  if (L.length >= L.listsize)
  { /* 当前存储空间已满,增加分配 */
    newbase=(ElemType *)realloc(L.elem,(L.listsize+LISTINCREMENT)*sizeof(ElemType));
    if (! newbase) exit(OVERFLOW);/* 存储分配失败 */
    L.elem = newbase;/* 新基址 */
    L.listsize += LISTINCREMENT;/* 增加存储容量 */
   }
  q = &(L.elem[pos-1]); /* q 指示插入位置 */
  for (p = &(L.elem[L.length-1]);p >= q;--p) *(p+1) = *p;
  /* 插入位置及之后的元素右移 *
   *q = e;/* 插入 e */
   ++L.length;/* 表长增 1 */
  return OK;
} /* ListInsert_Sq */
```

验证算法源程序中算法所对应的函数定义:

```
int ListInsert_Sq(SqList *L,int pos,ElemType e)
{
  ElemType *newbase, *p, *q;
```

```
  if (pos < 1 || pos > L->length+1) return (-1);
  if (L->length >= L->listsize)
  {
    newbase=(ElemType *)realloc(L->elem,(L->listsize+LISTINCREMENT)*sizeof(ElemType));
   if (! newbase) exit(-1);
   L->elem = newbase;
   L->listsize += LISTINCREMENT;
  }
 q = &((L->elem)[pos-1]);
 for (p=&((L->elem)[L->length-1]);p>=q;--p)
       *(p+1)=*p;
  *q = e;
  ++L->length;
  return (1);
}
```

(3) 算法描述的参数问题。

在类C语言算法(函数)中,在形参定义时,以"&"打头的参数作为引用参数,即在函数调用后发生改变的量,但在验证源程序的函数定义时,需将引用参数转换为指针类型,在函数定义中,把引用参数的使用转换为引用参数对象的使用。

例 1-4 顺序表的插入算法中的参数问题

类C语言描述:

```
Status ListInsert_Sq(SqList &L,int pos,ElemType e)
{
 if (pos < 1 || pos > L.length+1) return ERROR;
 if (L.length >= L.listsize)
 {
  newbase=(ElemType *)realloc(L.elem,(L.listsize+LISTINCREMENT)*sizeof(ElemType));
  if (! newbase) exit(OVERFLOW);
  L.elem = newbase;
  L.listsize += LISTINCREMENT;
 }
 q = &(L.elem[pos-1]);
 for (p = &(L.elem[L.length-1]);p >= q;--p) *(p+1) = *p;
 *q = e;
 ++L.length;
 return OK;
}
```

验证算法源程序中算法所对应的函数定义: