普通高等教育“十一五”国家级规划教材

李业丽 程晓锦 徐秀花 齐亚莉 编著

数据结构实验教程
（基于C语言）

21世纪计算机科学与技术实践型教程

丛书主编 陈明

清华大学出版社

普通高等教育“十一五”国家级规划教

纪计算机科学与技术实践型教程

李业丽 程晓锦 徐秀花 齐亚莉 编著

数据结构实验教程

（基于C语言）

丛书主编 陈明

清华大学出版社

内 容 简 介

数据结构是计算机学科的核心专业课程之一，它是软件开发的重要基础。为了配合数据结构课程的教学，加强读者对数据结构算法的理解，提高读者分析问题和解决问题的能力，本书根据数据结构课程教学内容，总结出每章的内容要点，有针对性地设计了一些数据结构实验，以加强基础实验的训练力度，起到举一反三的作用。对于每个实验，给出实验内容与要求、知识要点、实现提示、参考源程序及思考与提高，并在附录中给出了参考实验报告模板。

本书内容由浅入深、内容丰富、概念清楚、通俗易懂，特别注重对实际问题的分析和理解，具有较强的实用性。本书既可以作为高等院校各类相关专业本科生、专科生学习数据结构的上机实验指导，也可以作为相关专业自学考试、研究生入学考试、计算机技术与软件专业技术资格（水平）考试、计算机等级考试（三级或四级）应试复习资料，同时也可供各类学习数据结构的人员参考使用。

图书在版编目（CIP）数据

数据结构实验教程：基于C语言/李业丽等编著. --北京：清华大学出版社，2014
21世纪计算机科学与技术实践型教程
ISBN 978-7-302-34937-2

Ⅰ. ①数… Ⅱ. ①李… 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. ①TP311.12 ②TP312

中国版本图书馆CIP数据核字(2013)第321312号

责任编辑：谢　琛
封面设计：傅瑞学
责任校对：时翠兰
责任印制：宋　林

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：北京国马印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：21　　**字　　数**：484千字
版　　次：2014年4月第1版　　**印　　次**：2014年4月第1次印刷
印　　数：1～2000
定　　价：39.00元

产品编号：053016-01

《21世纪计算机科学与技术实践型教程》

编辑委员会

《21世纪计算机科学与技术实践型教程》

序

21世纪影响世界的三大关键技术：以计算机和网络为代表的信息技术；以基因工程为代表的生命科学和生物技术；以纳米技术为代表的新型材料技术。信息技术居三大关键技术之首。国民经济的发展采取信息化带动现代化的方针，要求在所有领域中迅速推广信息技术，导致需要大量的计算机科学与技术领域的优秀人才。

计算机科学与技术的广泛应用是计算机学科发展的原动力，计算机科学是一门应用科学。因此，计算机学科的优秀人才不仅应具有坚实的科学理论基础，而且更重要的是能将理论与实践相结合，并具有解决实际问题的能力。培养计算机科学与技术的优秀人才是社会的需要、国民经济发展的需要。

制订科学的教学计划对于培养计算机科学与技术人才十分重要，而教材的选择是实施教学计划的一个重要组成部分，《21世纪计算机科学与技术实践型教程》主要考虑了下述两方面。

一方面，高等学校的计算机科学与技术专业的学生，在学习了基本的必修课和部分选修课程之后，立刻进行计算机应用系统的软件和硬件开发与应用尚存在一些困难，而《21世纪计算机科学与技术实践型教程》就是为了填补这部分空白。将理论与实际联系起来，使学生不仅学会了计算机科学理论，而且也学会了应用这些理论解决实际问题。

另一方面，计算机科学与技术专业的课程内容需要经过实践练习，才能深刻理解和掌握。因此，本套教材增强了实践性、应用性和可理解性，并在体例上做了改进——使用案例说明。

实践型教学占有重要的位置，不仅体现了理论和实践紧密结合的学科特征，而且对于提高学生的综合素质，培养学生的创新精神与实践能力有特殊的作用。因此，研究和撰写实践型教材是必需的，也是十分重要的任务。优秀的教材是保证高水平教学的重要因素，选择水平高、内容新、实践性强的教材可以促进课堂教学质量的快速提升。在教学中，应用实践型教材可以增强学生的认知能力、创新能力、实践能力以及团队协作和交流表达能力。

实践型教材应由教学经验丰富、实际应用经验丰富的教师撰写。此系列教材的作者不但从事多年的计算机教学，而且参加并完成了多项计算机类的科研项目，他们把积累的经验、知识、智慧、素质融于教材中，奉献给计算机科学与技术的教学。

我们在组织本系列教材过程中，虽然经过了详细的思考和讨论，但毕竟是初步的尝试，不完善甚至缺陷不可避免，敬请读者指正。

本系列教材主编　陈明

2005年1月于北京

前　言

《数据结构》是计算机及相关专业的一门重要专业基础课程，也是一门必修的核心课程。在计算机科学的各领域中，都会用到各种不同的数据结构，学好数据结构这门课程，对从事计算机技术及相关领域的工作人员来说非常重要。

由于数据结构的原理和算法比较抽象，理解和掌握其中的原理就显得较为困难。学习这门课程，实验是非常关键的环节，上机实验是理解算法最佳的途径之一。为了帮助读者更好地学习本课程，理解和掌握算法设计所需的技术，作者通过多年的教学实践，收集、整理进而编写了本教程，希望读者通过上机实验加强对数据结构算法的理解，提高读者分析问题和解决问题的能力。

本书根据数据结构课程教学内容，总结出每章的内容要点，有针对性地设计了一些数据结构实验，对于每个实验，给出实验内容与要求、知识要点、实现提示、参考程序及思考与提高，所有的源程序都在 Turbo C 和 Visual C++ 6.0 环境下运行通过。通过这些实验，可以使读者了解并学会如何运用数据结构知识去解决现实世界中的一些实际问题，并具备设计较复杂算法的基本能力。在本书的附录中给出了参考实验报告模板，培养学生按照规范的形式书写实验报告的习惯。

本书一个重要的特点是根据学生的基础知识、兴趣爱好，将实验分成基础实验和提高实验。基础实验主要验证数据结构课程中的基本算法，练习巩固课程内容。提高实验设计了一些与实际问题紧密联系的难易程度不同的实验，读者可以根据自己的兴趣爱好与知识水平，自己选择实验题目。读者通过完成一系列的实验，巩固基础理论知识，培养分析、解决实际问题的能力，培养进行复杂问题程序设计的能力。

本书既可以作为高等院校各类相关专业本科生、专科生学习数据结构的上机实验指导，也可以作为相关专业自学考试、研究生入学考试、计算机技术与软件专业技术资格（水平）考试、计算机等级考试（三级或四级）应试复习资料，同时也可供各类学习数据结构的人员参考使用。选用该指导书上机实验的学校，可以根据学校自身的条件，在实验题目中有针对性地选一部分或全选。

本书是由北京印刷学院李业丽、程晓锦、徐秀花和齐亚莉编写，全书由程晓锦、李业丽负责统稿、定稿。在本书的编写过程中，得到了北京印刷学院计算机系老师的大力支持，在此深表感谢。

由于作者水平有限、时间仓促，本书难免存在一些缺点和错误，恳请读者及同行批评

指正。

为了方便教师教学和学生自学，本书配有电子教案和实验的全部程序，需要的读者可联系本书作者，电子邮箱：xjcheng@bigc.edu.cn。

编　者

2013年9月于北京

目　　录

第1章 线 性 表

1.1 内 容 要 点

线性表(Linear list)是最简单且最常用的一种数据结构。这种结构具有以下特点：存在一个唯一的没有前驱的(称为表头)数据元素；存在一个唯一的没有后继的(称为表尾)数据元素；此外，除表头外每一个数据元素均有一个直接前驱，除表尾外每一个元素均存在直接后继数据元素。

1.1.1 线性表的定义及其运算

1. 线性表的定义

线性表是 $n(n\geqslant 0)$个数据元素 $a_1,a_2,\cdots,a_n$ 组成的序列。其中 n 称为数据元素的个数或线性表的长度，当 $n=0$ 时称为空表，$n>0$ 时称为非空表。通常将非空的线性表记为 $(a_1,a_2,\cdots,a_n)$，其中数据元素 $a_i(1\leqslant i\leqslant n)$是一个抽象的符号，其具体含义在不同情况下是不同的，即它的数据类型可以根据具体情况而定。

2. 线性表的特征

从线性表的定义可以看出线性表的逻辑特征：

(1) 非空表有且仅有一个开始结点(表头结点)a_1，它没有直接前驱，可有一个直接后继；

(2) 非空表有且仅有一个终端结点(表尾结点)a_n，它没有直接后继，可有一个直接前驱；

(3) 其他结点都只有一个直接前驱和一个直接后继；

(4) 元素之间为一对一的线性关系。

3. 线性表的运算

常见线性表的运算有以下几种。

(1) 置空表(初始化) Init(L)：将线性表 L 置成空表。

(2) 求表长度 Length(L)：求给定线性表 L 的长度。

(3) 取表元素 Get(L,i,e)：若 1≤i≤length(L)，返回 TRUE，且 e 中存放第 i 个位置上的元素，否则返回 FALSE 且 e 中值无意义。

(4) 定位函数 LOCATE(L,X)：返回线性表 L 中第一个值为 X 的元素的位置，若没有则返回 0。

(5) 插入 Insert(L,X,i)：在线性表 L 中第 i 个数据元素之前插入一个值为 X 的新元素，如果插入成功($1 \leqslant i \leqslant length(L)+1$)，则返回 TRUE，否则返回 FALSE。

(6) 删除 Delete(L,i,e)：删除线性表 L 中第 i 个元素。当 $1 \leqslant i \leqslant len$ 时，返回 TRUE 表示删除成功，此时 e 中值为被删除的元素值，否则返回 FALSE 且 e 中值无效。

对线性表的操作还有很多，比如取前趋，取后继及排序等，但这些操作可以在以上基本操作的基础上完成。

1.1.2 线性表的顺序存储结构

线性表的顺序存储结构也称为顺序表。其存储方式为：在内存中开辟一片连续存储空间，该连续存储空间的大小要大于或等于顺序表的长度，然后让线性表中第一个元素存放在连续存储空间第一个位置，第二个元素紧跟着第一个元素之后，其余依此类推。可见，在线性表上，逻辑关系相邻的两个元素在物理位置上也相邻。

线性表顺序存储的特点：

很容易确定每个数据元素在存储单元中与起始地址的相对位置：假设线性表中元素为$(a_1,a_2,\cdots,a_n)$，设第一个元素 a_1 的内存地址为 $LOC(a_1)$，每个元素占 C 个存储单元，则第 i 个元素 a_i 的存储地址为 $LOC(a_i)=LOC(a_1)+(i-1)\times C$(其中 $1 \leqslant i \leqslant n$)。显然，对顺序表可以进行随机访问，故线性表的顺序存储结构是一种可以随机访问的存储结构。

1. 顺序表的结构

下面用 C 语言，给出顺序表的定义：

```
#define MAXSIZE  maxlen          /* maxlen 是一个无符号整数 */
enum enum{FALSE,TRUE};           /* 定义假(FALSE)真(TRUE)值 */
typedef  int ElemType;           /* 数据元素的类型,暂定义为 int,可用合适类型替换 */
typedef struct
{
    ElemType vec[MAXSIZE ];      /* 数据元素存储空间 */
    int len;                     /* 顺序表的长度 */
} SequenList;
```

注：ElemType 为线性表中数据元素的数据类型，此处为了简单而取 int 类型，在实际应用中应该用实际的类型替换。

有了线性表的顺序存储结构后，下面就可以讨论如何在这种结构上实现有关数据元素的运算问题。这里重点讨论线性表元素的插入和删除。

2. 顺序表的基本运算

以下操作中，假设线性表的长度为 n。

1) 插入运算

线性表的插入是指在表的第 i 个位置上插入一个值为 x 的新元素，通常在第 i 个位置($1 \leqslant i \leqslant n+1$)插入一个新元素时，需要有 $n-i+1$ 个元素进行移动。通过以下步骤完成顺序表的插入运算：

(1) 将 $a_n \sim a_i$ 共 $n-i+1$ 个元素依次由后向前向后移动到下一个位置，为新元素让

出位置；

(2) 将 x 置入空出的第 i 个位置；

(3) 修改 len 值(即修改表长)，使之加 1。

算法描述如下：

```
enum BOOLEAN  InsertInSeqlist(SequenList * L, int  i, ElemType x)
{
    int j;
    if(L->len>=MAXSIZE)
    {
        printf("线性表溢出!\n");
        return FALSE;
    }
    if(i<1 || i>L->len+1)
    {
        printf("插入位置错误!\n");
        return FALSE;
    }
    for(j=L->len-1;j>=i-1;j--)
        L->vec[j+1]=L->vec[j];          /*元素后移*/
    L->vec[i-1]=x;                      /*插入元素 x*/
    L->len++;                           /*表长度增加 1*/
    return TRUE;
}                                       /*插入操作结束*/
```

性能分析：

插入算法花费的时间，主要在元素的移动上(其他语句花费的时间可以省去)，即从插入位置到最后位置的所有元素都要后移一位，使空出的位置插入值为 x 的元素。但是，插入的位置是不固定的，当插入位置 $i=1$ 时，全部元素都得移动，需 n 次移动，当 $i=2$ 时，移动 $n-1$ 个元素，当 $i=n$ 时，仅需移动元素一次，一般地，需要移动 $n-i+1$ 个元素。设 p_i 为在第 i 个数据元素之前插入一个元素的概率，假设在顺序表的任何位置上插入元素都是等概率的，即 $p_i=1/(n+1)$，则在长度为 n 的顺序表中插入一个元素时所需的平均移动次数为：

$$\sum_{i=1}^{n+1}\frac{1}{n+1}(n-i+1)=\frac{n}{2}$$

可见在顺序表上插入操作平均需要移动表中一半的数据元素，即时间复杂度为 $O(n)$。

2) 删除运算

线性表的删除运算是指将表中第 i 个元素从线性表中去掉，i 的取值范围为 $1\leqslant i\leqslant n$。通过以下步骤完成顺序表的删除运算：

(1) 将 $a_{i+1}\sim a_n$ 顺序依次由前向后向前移动一个位置；

(2) 修改 len 值(即修改表长)，使之减 1。

算法描述如下：

```
enum BOOLEAN DeleteFromSeqlist(SequenList * L, int  i , ElemType * e)
{
    if(i<1 || i>L->len){
        printf("删除位置错误!\n");
        return FALSE;                    /* 欲删除元素的位置错误 */
    }
    L->len--;i--;
    while(i<L->len)
    {
        L->vec[i]=L->vec[i+1];           /* 元素前移 */
        i++;
    }
    return TRUE;
}                                        /* 删除操作结束 */
```

性能分析：

与插入运算相同，删除第 i 个元素时，后面的元素 $a_{i+1}\sim a_n$ 都要向前移动一个位置，共移动了 $n-i$ 个元素，故平均移动次数：

$$\sum_{i=1}^{n}\frac{1}{n}(n-i)=\frac{n-1}{2}$$

这说明在顺序表上作删除运算大约需要移动表中一半的元素，显然该算法的时间复杂度为 $O(n)$。

1.1.3 线性表的链式存储结构

线性表的链式存储结构，也称为链表。链式存储是用一些任意的存储单元来存储线性表中的元素，这些单元在物理上可以是连续的，也可以是不连续的。为了能正确地描述元素之间的逻辑关系，除了存储元素本身的信息外，还需存储指示元素之间逻辑关系的信息。这两部分信息组成数据元素 a_i 的存储映像，称为结点。

在链表中，每个结点所占的存储空间分为两部分：存放数据元素值的域称为数据域，存放结点之间相互关系的域称为指针域或链接域。在定义的链表中，若指针域中只存储一个指针指向下一个元素地址，则称这样的链表为单链表或线性链表。

1. 单链表结构

线性链表中的结点结构可描述为：

Data	next

其中 Data 域用来存放结点数据元素信息，类型由具体问题而定，next 域用来存放下一个元素地址，即存储指示其直接后继的指针。单链表结构如图 1-1 和图 1-2 所示，其假设每个元素结点占 10 个存储单元。

由图 1-2 可见，用单链表表示线性表时，数据元素之间的逻辑关系是通过链表结点中

头指针		地址	data域	next域
head	150	110	a_2	180
		120		
		130	a_4	170
		140	a_6	NULL
		150	a_1	110
		160		
		170	a_5	140
		180	a_3	130

图 1-1　单链表存储示意图

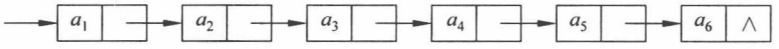

图 1-2　单链表的逻辑表示

的指针反映出来的，即指针是数据元素之间逻辑关系的映像。

2. 单链表上的基本运算

为了便于实现各种运算，通常在单链表的第一个结点前增设一个附加结点，称为头结点，它的结构与表结点相同，其数据域可不存储信息，也可存储如线性表长度一类的附加信息。单链表设置头结点的作用归纳如下：

(1) 由于一个线性链表的头结点固定不变，头结点的 next 域指向线性链表中的第一个元素，因此指向带头结点的线性链表的指针是不变的，为有关线性链表的操作带来方便。

(2) 头结点可以认为是原线性链表中第一个元素的直接前驱，因此无论是插入操作还是删除操作皆不必考虑当前结点是否为头结点，方便了插入删除操作。

用 C 语言描述结点结构如下：

```
#define Node LinkListNode
typedef  int ElemType;              /* 数据元素的类型,暂定义为 int,可用其他类型替换 */
enum BOOLEAN {FALSE,TRUE};
typedef struct Node
{
    ElemType data;                  /* 数据域 */
    struct Node *next;              /* 指针域 */
} *LinkList;
```

1) 单链表的构建

链表与顺序表不同，它是一种动态管理的存储结构，链表中的每个结点占用的存储空间不是预先分配，而是运行时系统根据需求而生成的，因此建立单链表从空表开始，每读入一个数据元素则申请一个结点，然后插在链表的尾部。

算法描述：

```
LinkList  CreatList(int n)
```

```
{
    ElemType     x;                                          /*设数据元素的类型为 int*/
    int     k;
    struct Node   *head, *r, *p;
    p=(struct Node *)malloc(sizeof(struct Node));  /*构建头结点空间*/
    if(p==NULL) return NULL;                                 /*内存溢出*/
    head=p;p->next=NULL;     r=p;
    for(k=1;k<=n;k++)                                        /*循环构建 n 个结点*/
    {
        printf("请输入第%d个元素:\n",k);
        scanf("%d",&x);
        p=(struct Node *)malloc(sizeof(struct Node));
        if(p==NULL) {
            DestroyList(head);                               /*溢出,删除已建立的线性表*/
        return NULL;
        }
        p->data=x;
        p->next=NULL;
        r->next=p;
        r=r->next;
    }
    return head;
}                                                            /*单链表建立操作结束*/
```

说明：

(1) 每次动态申请内存后，需要验证申请是否成功，如果申请失败，应该释放已分配的内存，只有申请成功时，才可进行下面的操作。

(2) 如果 n 取 0，则为建立一个空线性链表。

2) 单链表上元素的查找

从头指针开始，一般有两种查找方法：按元素的序号和按某个给定值。

(1) 按序号查找

设单链表的长度为 n，查找表中第 i 个结点的算法的基本思想是：从线性表中的第一个结点开始顺序扫描链表中的每一个结点，如果扫描到第 i 个结点或扫描的结点为空，则扫描结束。如果当前结点非空，则为查找的结点。用指针 p 指向当前扫描到的结点，首先将 p 置链表头结点的下一个结点，扫描前先将 i 减一，如果 i 值非 0 且 p 非空，则将 p 移到下一个结点。

算法描述：

```
struct Node    * FindInListByPosition (LinkList list, int i)
{
    struct Node * p=list->next;
    while(--i!=0 && p!=NULL)
```