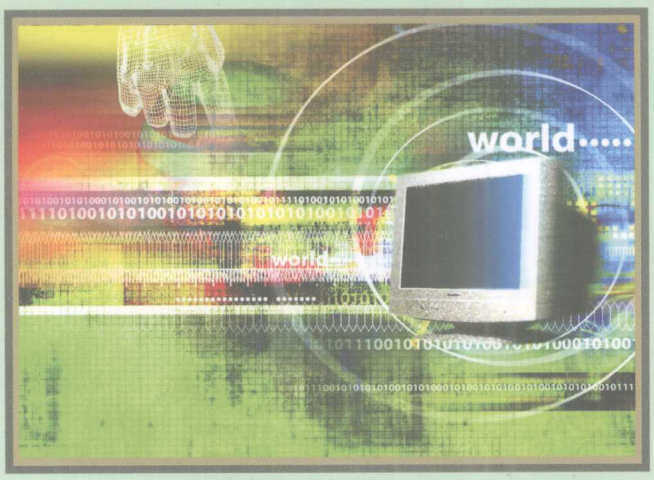
高等院校计算机专业优秀基础课教材

数据结构实践训练教程

ShuJu JieGou ShiJian XunLian JiaoCheng

刘光然 主编
徐棣 罗梅 李典蔚 编著



南开大学出版社

数据结构实践训练教程

主编　刘光然
编著　徐棣　罗梅　李典蔚

南开大学出版社
天　津

图书在版编目(CIP)数据

数据结构实践训练教程 / 刘光然主编. —天津：南开大学出版社，2009.4

ISBN 978-7-310-03113-9

Ⅰ.数… Ⅱ.刘… Ⅲ.数据结构—教材 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字(2009)第 035431 号

南开大学出版社出版发行

出版人：肖占鹏

地址：天津市南开区卫津路 94 号 邮政编码：300071

营销部电话：(022)23508339 23500755

营销部传真：(022)23508542 邮购部电话：(022)23502200

*

南开大学印刷厂印刷

全国各地新华书店经销

*

2009 年 4 月第 1 版 2009 年 4 月第 1 次印刷

787×1092 毫米 16 开本 17.75 印张 445 千字

定价：34.00 元(含光盘一张)

如遇图书印装质量问题，请与本社营销部联系调换，电话：(022)23507125

内容提要

本书深入浅出地阐述数据结构的基础知识，并根据每章的知识点，精选出具有针对性、实用性、普及性的经典实训项目，让学生在训练过程中边学边练，在不知不觉中得以全面提高计算机素质。最后还给出了3个用以训练学生综合运用能力的综合项目案例。

本书的特点是“以能力培养为核心，以技能训练为主线，以实践项目为载体，以理论知识为支撑”；注重理论和实践相结合，用理论指导实践，在实践中理解并运用理论；保证各个实训项目的科学性、实践性、实用性和可操作性，附赠一张CD-ROM配套程序光盘。

本书既可作为高等学校应用型本科和高等职业院校计算机相关专业学生的实训教材，也可作为计算机程序爱好者的自学参考书。

前　言

在大力发展职业教育的今天，各高校对学生实际技能的培养越来越重视，尤其是应用型本科学校和高等职业院校更是把培养学生的动手能力和应有的职业技能作为首要职责，于是教学改革不断深入。随着 CDIO 教学模式的引进和推广，越来越多的学者开始进行研究和尝试，基于项目的教学培养方式得到普遍认可并获得广泛的应用，基于项目的实践教学已经成为目前各高校赖以培养学生实际操作技能的主要法宝之一。基于此，结合多年实际教学经验，我们尝试着编写了本教程。

在本教程的编写过程中，我们遵循“以能力培养为核心，以技能训练为主线，以实践项目为载体，以理论知识为支撑”的思想；注重理论和实践相结合，用理论指导实践，在实践中理解并运用理论，保证各个实训项目的科学性、实践性、实用性和可操作性。

本教程共计 8 章。第 1 章介绍线性表的概念，提供了学生成绩管理系统、考试报名管理系统和约瑟夫生者死者游戏等 4 个项目案例；第 2 章介绍栈和队列的概念，提供了勇闯迷宫游戏、N 皇后问题和停车场管理系统等 3 个项目案例；第 3 章介绍串的概念，给出了关键字检索和四元线性方程组求解等 2 个实训案例；第 4 章介绍树和二叉树的概念，列举了家谱管理、表达式求值、图像压缩编码优化等 3 个实际问题及其解决方案；第 5 章介绍了图的基本知识，给出了公交路线管理模拟、导航最短路径查询、电网建设造价模拟、软件工程进度规划等日常实用案例；第 6 章介绍了查找的基本概念，给出了顺序查找、折半查找、二叉排序树和哈希查找等 4 个简单案例；第 7 章介绍了排序的有关概念，并将相关的排序算法融为一个大的程序，即各种排序算法的比较程序；第 8 章是综合应用篇，我们给出了迷宫益智游戏、景区旅游信息管理系统（Console 版本）、景区旅游信息管理系统（MFC 版本）等 3 个要求利用所学知识完成的综合性实训案例，以提高学生的综合运用所学知识的实际能力。

在学习本教程时，需要读者具有一定的 C++编程基础和阅读程序代码的能力。

本教程由刘光然总体策划；各章节基本概念的内容由徐棣和罗梅确定，由罗梅老师主笔；实训案例由徐棣和李典蔚设计，程序源代码由李典蔚老师编写和调试（详见本书配套的 CD-ROM 光盘）；全书最后由刘光然、徐棣老师统一修改定稿。

本教程在编写过程中，作者参阅了相关的书籍和网络资源，在此向所有这些书籍、资料的作者们表示衷心的感谢，同时感谢我的朋友和同事以及张燕老师、尹建国老师对本教程的写作和出版提供的帮助。

读者在学习过程中如需与作者联系，请发邮件至 Liuguangran@163.com。由于作者的能力有限，水平不高，错误之处在所难免，恳请读者朋友批评指正。

刘光然

2009 年元旦于天津

目 录

第一章　线性表

线性表是数据结构中最简单、最常用的一种线性结构，也是学习数据结构全部内容的基础，其掌握的好坏直接影响着后继知识的学习。本章通过四个模拟实例来学习线性表的顺序和链式存储结构，首先通过使用有关数组的操作实现学生成绩管理，其次通过使用有关线性链表的操作实现考试报名管理，然后通过使用循环链表的操作实现约瑟夫生者死者游戏。

1.1 实践目的和要求

1.1.1 实践目的

1）掌握线性表顺序存储结构和链式存储结构的思想和特点；
2）掌握上机调试线性表的基本方法；
3）掌握线性表的插入、删除、查找以及线性表合并等运算在顺序存储结构和链式存储结构上的实现。

1.1.2 实践要求

1）认真阅读和掌握本章的程序，注意插入、删除时元素的移动原因、方向及先后顺序，理解不同的函数形参与实参的传递关系。
2）上机运行本章的程序，保存和打印出程序的运行结果，并结合程序进行分析。
3）按照你对线性表的操作需要，重新改写程序并运行。
4）重点理解链式存储的特点及指针的含义。
5）注意比较顺序存储与链式（单向、双向、循环链表）存储的特点及实现方法。
6）注意比较带头结点、无头结点链表实现插入、删除运算时的区别。
7）单向链表的操作是数据结构的基础，要注意对这部分的常见算法的理解。

1.2 基本概念

1.2.1 线性表的定义

线性表是具有相同数据类型的数据元素的一个有限序列。该序列中所含元素的个数叫作线性表的长度，用 n（n≥0）表示。当 n=0 时，表示线性表是一个空表，即表中不含任何元素。设序列中第 i（i 表示位序）个元素为 e_i（1≤i≤n），则线性表的一般表示为：（e_1，e_2，…，e_{i-1}，e_i，e_{i+1}，…，e_n），其中：e_1 为第一个元素，又称表头元素，e_2 为第二个元素，e_n 为最后一个元素，又称表尾元素。对于非空线性表，除第一个元素之外，表中的每个元素均有且只有一个直接前驱；除最后一个元素之外，表中每个元素均有且只有一个直接后继。

一个线性表可以用一个标识符来命名，如用 L 来命名上面的线性表，则有：

$L=(e_1, e_2, \cdots, e_{i-1}, e_i, e_{i+1}, \cdots, e_n)$

线性表最重要的性质是线性表中的元素在位置上是有序的，即第 i 个元素 e_i 处在第 i-1 个元素 e_{i-1} 的后面和第 i+1 个元素 e_{i+1} 的前面，这种位置上的有序性就是一种线性关系，所以线性表是一个线性结构，用二元组表示为：

L=（D，R）

D={e_i|1≤i≤n，n≥0，e_i 属于 ElemType 类型}　//ElemType 是 C/C++的类型标识符

R={r}，r={<e_i，e_{i+1}>|1≤i≤n-1}

当一个线性表的元素升序或降序排列时，称为有序线性表，简称为有序表。

有多种存储方式能将线性表存储在计算机内，其中最常用的是顺序存储结构和链式存储结构，分别简称为顺序表和链表。

1.2.2 线性表的顺序存储结构

顺序存储结构是最简单的存储方式，把线性表中的所有元素按照其逻辑顺序依次存储到计算机存储器中的一块连续的存储空间中。通常使用一个足够大的数组，从数组的第一个元素开始，将线性表的结点依次存储在数组中。假设线性表的元素类型为 ElemType，线性表长度为 n，则整个线性表所占用存储空间的大小为 n* sizeof（ElemType）。

在定义一个线性表的顺序存储结构时，需要定义数组来存储线性表中的所有元素和定义一个整型变量来存储线性表的长度。

○顺序存储结构的特点

逻辑关系上相邻的两个元素在内存中的物理位置上也相邻。

○顺序存储结构的优点

①无需为表示结点间的逻辑关系而额外增加存储空间；

②能随机访问线性表中的任一元素。线性表的第 i 个元素 e_i 的存储地址可以通过以下公式求得：LOC（e_i）=LOC（e_1）+（i-1）*sizeof（ElemType），其中 LOC（e_1）是线性表的表头元素 e_1 的存储位置，通常称作线性表的起始位置或基地址。

○顺序存储结构的缺点

①由于要求占用连续的存储空间，存储分配只能预先进行，线性表的最大长度需预先确定，从而浪费大量的存储空间；

②插入与删除运算的效率很低。为了保持线性表中的元素的顺序，执行线性表的插入和删除操作时需要移动其他大量元素，这对于插入和删除操作频繁的线性表及每个元素所占字节较大的问题将导致系统的运行速度难以提高；

③线性表的顺序存储结构的存储空间不便于扩充。当一个线性表按顺序存储结构存储后，如果在原线性表的存储空间后找不到与之连续的可用空间，但还需要插入新的元素，则会发生“上溢”错误。

1.2.3 线性表的链式存储结构

链式存储结构即用一组任意的存储单元存储线性表中的元素，每个存储结点不仅包含所存储元素本身的信息即数据域，而且包含元素之间的逻辑关系的信息（前驱结点包含后继结

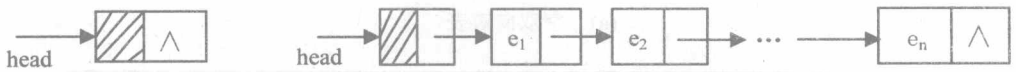
点的地址信息）即指针域。一般地，每个存储结点有一个或多个这样的指针域。若一个结点中的某个指针域不需要指向任何结点，则将它的值置为空指针，用常量 NULL 表示。

根据指针域的不同和结点构造指针域方法的不同，链式存储结构有如下三种形式（每种形式均分为带头结点结构和不带头结点结构两种结构）。

1.2.3.1 单链表（线性链表）

链表的每个存储结点除要存储线性表元素本身的信息外，只设置一个指针域用来指向其直接后继结点。

如图 1.1 所示为带头结点的单链表结构。



(a) 空单链表　　(b) 非空的单链表

图1.1　单链表的存储结构

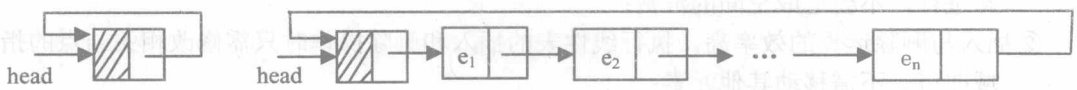
○带头结点的单链表的特点

每个单链表都有一个头指针，整个链表的存取必须从头指针开始，头指针指向头结点的位置，头结点的指针域指向第一个元素的位置，最后的结点指针为空。当链表为空时，头结点的指针为空值；链表非空时，头结点指向第一个结点。

1.2.3.2 循环链表

循环链表是另一种形式的链式存储结构，是单链表的变形。它的特点是表中最后一个结点的指针域指向头结点，整个链表形成一个环。因此，从表中任意一个结点出发都可以找到表中的其他结点。

如图 1.2 所示为带头结点的循环链表结构。



(a) 空循环链表　　(b) 非空的循环链表

图1.2　循环链表的存储结构

循环链表和单向链表基本相同，差别仅在于算法中判断链表是否已到达末结点的条件不是结点的指针是否为空，而是它的指针是否等于头指针，即循环单向链表末结点的指针不为空，而是指向了表的前端。

○循环链表的特点

只要知道表中某一结点的地址，就可搜寻到所有其他结点的地址。

1.2.3.3 双向链表

双向链表是线性表的另一种形式的链式存储结构，双向链表的存储结点中除要存储线性表元素本身的信息外，还设置有两个指针域，分别指向其直接后继结点和直接前驱结点。双向链表克服了单链表的单向性的缺点。

如图 1.3 所示为带头结点的双向链表结构。

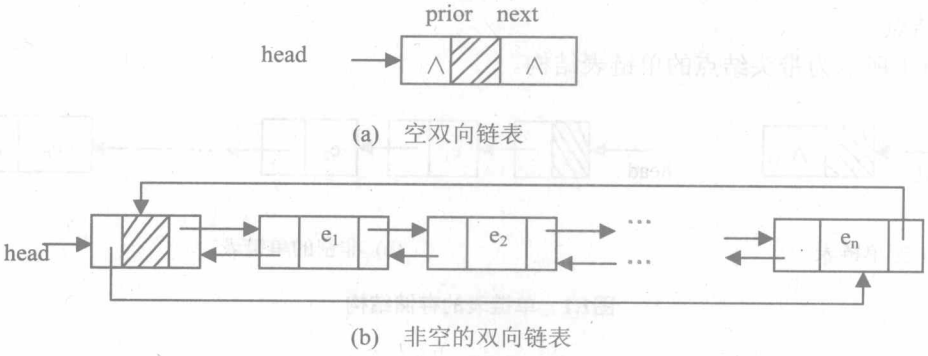


图1.3 双向链表的存储结构

双向链表也有循环链表形式，即链表中存在两个环。一个结点的直接前驱的直接后继和该结点的直接后继的直接前驱都是指向该结点的，即假定某存储结点的指针域为 p，则有：

$$p == p->prior->next == p->next->prior$$

○链式存储结构的特点

逻辑关系上相邻的两个元素在内存中的物理位置通过指针来表示。

○链式存储结构的优点

①由于不要求占用连续的存储空间，不需预先确定线性表的最大长度，存储分配可以动态进行，不会造成空间的浪费；

②插入与删除运算的效率高。执行线性表的插入和删除操作时只需修改相关结点的指针域即可，不需移动其他元素；

③线性表的链式存储结构的存储空间便于扩充。由于不要求存储空间的连续，因此只要内存空间还有剩余就可以插入新的元素。

○链式存储结构的缺点

①由于要存储相邻结点的地址信息，所以存储空间的开销较大；

②不具有顺序存储结构的随机存取特点，访问任何元素均需从头结点开始，影响访问效率。

1.2.4 线性表的基本运算

线性表是一个相当灵活的数据结构，它的长度可以根据需要增长或缩短。设基本线性表 $L=(e_1, e_2, \cdots, e_{i-1}, e_i, e_{i+1}, \cdots, e_n)$，对基本线性表 L 的基本运算如下：

1）InitList（&L）初始化：构造一个新的空线性表 L。

2）DestroyList（&L）撤销：销毁一个已有线性表 L。

3）ClearList（&L）清空：将一个已有线性表 L 置为空表。

4）ListEmpty（L）判空：判断一个已有线性表 L 是否为空表，是空表则返回 TRUE，否则返回 FALSE。

5）ListLength（L）求表长：返回一个已有线性表 L 中元素的个数。

6）GetElem（L，i，&e）取元素：返回一个已有线性表 L 中的第 i（1≤i≤ListLength（L））个元素的值。

7）LocateElem（L，e，compare（））元素定位：返回 L 中第一个与 e 满足关系 compare（）的数据元素的位置，若不存在这样的数据元素则返回 0。

8）PriorElem（L，cur_e，&pre_e）取直接前驱：若线性表 L 中存在元素 cur_e，且不是表头元素，则用 pre_e 返回它的直接前驱，否则操作失败。

9）NextElem（L，cur_e，&next_e）取直接后继：若线性表 L 中存在元素 cur_e，且不是表尾元素，则用 next_e 返回它的直接后继，否则操作失败。

10）ListInsert（&L，i，e）插入元素：在线性表 L 的第 i（1≤i≤ListLength（L）+1）个元素之前插入数据元素 e。线性表 L 长度加 1。

11）ListDelete（&L，i，&e）删除元素：删除线性表 L 中的第 i（1≤i≤ListLength（L））个元素。线性表 L 的长度减 1。

12）ListTraverse（L，visit（））遍历：对给定线性表 L 中的每一个数据元素依次调用 visit（）进行遍历。

13）CopyList（L，C）复制：将给定线性表 L 复制为线性表 C。

14）Merge（A，B，C）合并：将给定的线性表 A 和 B 合并为线性表 C。

根据存储方式的不同，上述运算的实现方法也不一样。

1.3 实践案例

1.3.1 学生成绩管理系统

1.3.1.1 系统简介

学生成绩管理是学校教务部门日常工作的重要组成部分，其处理信息量很大。本项目是对学生成绩管理的简单模拟，用控制台选项的选择方式完成下列功能：输入学生数据；输出学生数据；学生数据查询；添加学生数据；修改学生数据；删除学生数据。

1.3.1.2 设计思路

本项目的实质是完成对学生成绩信息的建立、查找、插入、修改、删除等功能。项目在设计时应首先确定系统的数据结构，定义类的成员变量和成员函数；然后实现各成员函数以完成对数据操作的相应功能；最后完成主函数以验证各个成员函数的功能并得出运行结果。

○数据结构

分析发现，本项目的数据是一组学生的成绩信息，每条学生的成绩信息可由学号、姓名和成绩组成。而且这组学生的成绩信息具有相同特性，属于同一数据对象，相邻数据元素之

间存在序偶关系。由此可以看出，这些数据具有线性表中数据元素的性质，所以该系统的数据适合采用线性表来存储。

线性表有两种存储方式，顺序表是线性表的顺序存储结构，是指用一组连续的内存单元依次存放线性表的数据元素。在顺序存储结构下，逻辑关系相邻的两个元素在物理位置上也相邻，这是顺序表的特点。顺序表适宜于作查找这样的静态操作，其优点是存储密度大，存储空间利用率高，缺点是插入或删除元素时不方便。

本项目确定使用线性表的顺序存储结构。

由于C语言的数组类型也有随机存储的特点，一维数组的机内表示就是顺序结构。因此，本项目可用C语言的一维数组实现线性表的顺序存储。

本系统的数据结构决定了其使用时的特性，根据顺序存储的特点可以发现，本系统可以方便的随机存取表中任一元素O（1），存储空间使用紧凑；缺点是在插入，删除某一元素时，需要移动大量元素O（n），预先分配空间需按最大空间分配，利用不充分，数组容量难以扩充。

○程序设计和头文件定义

经过上述分析，可确定系统的数据结构为数组，数组中的元素是学生成绩信息类。Student类的成员变量和成员函数在头文件Student.h中定义，具体描述为：

```
class Student
{
public:
    void    Creat   ( Student stu[] );
    void    Insert  ( Student stu[] );
    void    Delete  ( Student stu[] );
    void    Lookup  ( Student stu[] );
    void    Update  ( Student stu[] );
    void    Stat    ( Student stu[] );
    int     Length  ( Student stu[] );
    void    Print   ( Student stu[] );

    Student();

private:
    string  name;                   //姓名
    long    num;                    //学号
    float   score;                  //成绩
};
```

1.3.1.3 程序清单

```
/*****************************************************************************
    包含输入输出头文件 iostream 和字符串头文件 string，下面的案例将不再重复包含
*****************************************************************************/
```

```
#include "iostream"
#include "string"
using namespace std;

#include "Student.h"

/*****************************************************************************
    Student 类的构造函数，初始化元素数据
*****************************************************************************/
Student::Student()
{
    name = '?';
    num = 0;
    score = 0;
}

/*****************************************************************************
    创建学生成绩信息的线性表（数组）
*****************************************************************************/
void Student::Creat(Student stu[])
{
    cout << "请输入学生人数：" ;
    int n;
    cin >> n;
    cout<<"姓名"<<'\t'<<"学号"<<'\t'<<"成绩"<<endl;
    for(int i = 1; i <= n; i ++)
    {
        string newname;
        long newnum;
        float newscore;
        cin >> newname;
        stu[i].name = newname;
        cin >> newnum;
        stu[i].num = newnum;
        cin >> newscore;
        stu[i].score = newscore;
    }
}
```

```
/*****************************************************************************
    输出线性表中所有的学生成绩信息
*****************************************************************************/
void Student::Print(Student stu[])
{
    cout << "姓名" << '\t' << "学号" << '\t' << "成绩" << '\n';
    int i = 1;
    while (stu[i].num) {
        cout << stu[i].name << '\t' << stu[i].num << '\t' << stu[i].score << '\n';
        i++;
    }
    cout << '\n';
}

/*****************************************************************************
    在线性表的某个位置上插入学生信息
*****************************************************************************/
void Student::Insert(Student stu[])
{
    if (Student::Length(stu) == 100) {
        cout << "存储空间已满，不能进行插入操作！" << '\n';
    }
    else
    {
        cout << "请输入要插入的位置：";
        int m;
        cin >> m;
        int n = Student::Length(stu);
        if (m > n + 1) {
            cout << "插入位置不正确，请重新输入！" << '\n';
            Student::Insert(stu);
        }
        else {
            for(int i = n; i >= m; i --)                    //数组中的数据依次后移
            {
                stu[i + 1].name = stu[i].name;
                stu[i + 1].num = stu[i].num;
                stu[i + 1].score = stu[i].score;
            }
            cout << "请依次输入姓名，学号，成绩" << '\n';
```

```
                string newname;
                long newnum;
                float newscore;
                cin >> newname;
                stu[m].name = newname;
                cin >> newnum;
                stu[m].num = newnum;
                cin >> newscore;
                stu[m].score = newscore;
            }
        }
}

/*****************************************************************************
    根据学号删除线性表中对应的学生信息
*****************************************************************************/
void Student::Delete(Student stu[])
{
    cout << "请输入你要删除的学号：" ;
    long num;
    cin >> num;
    int i = 1;
    if (!stu[i].num) {
        cout << "你要删除的学号不存在，请重新输入！" << '\n';
        Student::Delete(stu);
    }
    while (stu[i].num) {
        if (stu[i].num == num) {
            int n = Student::Length(stu);
            for(int j = i; j < n; j ++)                    //数组中数据依次前移
            {
                stu[i].name = stu[i + 1].name;
                stu[i].num = stu[i + 1].num;
                stu[i].score = stu[i + 1].score;
            }
            stu[j].name = '?';
            stu[j].num = 0;
            stu[j].score = 0;
            break;
        }
```