普通高等教育“十一五”规划教材
高等学校计算机科学与技术系列教材

# 数据结构
## 实验指导与课程设计教程

陈建新　李志敏　主编



科学出版社
www.sciencep.com

通高等教育“十一五”规划教材
高等学校计算机科学与技术系列教材

# 数据结构实验指导与课程设计教程

陈建新　李志敏　主编

科学出版社
北京

## 内 容 简 介

本书是《数据结构》一书配套的实验教材，用于辅助实验教学。全书共分三篇。第一篇为基础实验，介绍数据结构与算法基础知识的实验，包括线性表、栈和队列、串、数组，以及广义表、树和二叉树、图、查找、排序等内容，一共有12个实验。第二篇为综合实验，是数据结构知识的应用与提高，包括链表的应用，栈和队列的应用，树结构的应用，图结构的应用以及文本文件检索等综合性实验内容，共10个实验。第三篇为课程设计，详细介绍了7个课程设计的课题，综合性较强，另外还给出了一部分实训项目，内容涉及数据结构课程的多个应用领域，以引导学生进行开发实践。

本书既可以作为《数据结构》课程的实验教材，也可作为其课程设计的参考用书。

**图书在版编目(CIP)数据**

数据结构实验指导与课程设计教程/陈建新，李志敏主编. —北京：科学出版社，2010.2

普通高等教育“十一五”规划教材

(高等学校计算机科学与技术系列教材)

ISBN 978-7-03-026571-5

Ⅰ.①数… Ⅱ.①陈…②李… Ⅲ.①数据结构—高等学校—数学参考资料 Ⅳ.①TP311.12

中国版本图书馆CIP数据核字(2010)第017456号

责任编辑：黄金文/责任校对：翟 薯

责任印制：彭 超/封面设计：苏 波

科学出版社 出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

武汉市新华印刷有限责任公司印刷

科学出版社发行 各地新华书店经销

*

2010年1月第 一 版 开本：787×1092 1/16

2010年1月第一次印刷 印张：15 3/4

印数：1—3 000 字数：357 000

**定价：24.00元**

(如有印装质量问题，我社负责调换)

# 本书编委会


**主　编**　陈建新　李志敏

**编　委**　（按姓氏拼音为序）

陈佛敏　桂　超　李国屏　申鼎才

王　邯　许中元　张凯兵　张万山

周天宏　周云才

# 前　言

数据结构是计算机科学与技术及其相关专业的一门理论性较强的基础课，又是一门实践性较强的专业技术课。它是算法设计与分析、操作系统、软件工程、数据库概论、编译原理、计算机图形学等专业课程的基础课。它所研究的问题是计算机程序设计中的数据元素、数据对象之间的关系以及非数值计算的数据处理问题。数据的操作主要包括查找、插入、删除和遍历等非数值型计算，如何在适当的存储结构下实现这些操作算法是数据结构研究的核心问题。不同的数据结构所能施加的运算不同，不同的存储结构直接影响运算算法的实现和效率。数据的运算定义取决于逻辑结构，数据的运算算法依赖于存储结构。透彻掌握数据结构的理论与方法，有助于合理地组织存储数据、设计高效的算法、编写高质量的程序，满足实际应用的需要。

在数据结构教学与学习的过程中，实践能力和技巧的训练是一个重要的环节。为了配合《数据结构》课程教学需要，帮助和指导学生提高实践应用能力，我们组织编写了这本《数据结构实验指导与课程设计教程》，作为学习的辅助教材。

本教材共分三篇。第一篇为基础实验，介绍数据结构与算法基础知识和实验，包括线性表、栈和队列、串、数组，以及广义表、树和二叉树、图、查找、排序等内容，一共有 12 个实验，可以帮助学生熟练掌握基础知识和基本算法；第二篇为综合实验，是数据结构知识的应用与提高，包括链表的应用，栈和队列的应用，树结构的应用，图结构的应用以及文本文件检索等综合性实验内容，共有 10 个实验，用以培养学生分析和解决实际问题的能力；第三篇为课程设计，详细介绍了 7 个课程设计的课题，综合性比较强，精选实际应用课题，引导学生进行开发实践。内容涉及数据结构课程的多个应用领域，供各类教学和学习者参考。

与其他实践性教程相比，本教材有如下特色：

1. 教材努力作到“门槛低，坡度缓，层次高”。在内容编排上，先从验证型实验出发，引导学生独立进行综合实验设计，最后达到能完成综合课程设计的目标。

2. 本书内容丰富、实用性强、适用面广。既可作为《数据结构》教材的学习参考书和实验指导书，又可供高等院校各专业学生学习、实验、课程设计和考前复习，还可供教师和其他专业技术人员参考。

3. 根据数据结构教学大纲精心选择基础实验内容。考虑到本课程开课时间一般在本科低年级，学生编程能力不是很强，对基础实验编写做到过程描述详细，代码注释完整，便于初学者模仿训练，循序渐进，稳步提高。

4. 综合实验的内容按课程教学顺序设计。同时考虑到实际应用的要求，使综合实验既巩固大纲要求的知识点，又接近课程设计项目的需要，循序渐进训练学生的分析问题和解决问题以及编程能力。

5. 课程设计项目选题新颖，应用性强。将课程设计过程进行模块化描述。每个项目

都有设计要求，需求分析，算法设计，编程和实例测试。通过详细的实例分析，锻炼学生动手能力，引导学生完成课程设计。

6. 本教程配套学习资源丰富，C 语言程序源代码完整，可以直接编译运行。

本书由陈建新、李志敏组织编写。参加本书编写和讨论的有陈佛敏、桂超、李国屏、申鼎才、王邯、许中元、张凯兵、张万山、周天宏、周云才等。

本书的出版得到了孝感学院的大力支持。在本书的写作过程中，参考了相关教材及网络资料，在此一并致谢。由于作者水平有限，书中难免存在一些缺点和错误，殷切希望广大读者及同行批评指正。

本书的辅助教学资源可以从 http://www.xgu.edu.cn/精品课程网页下载（正文中简称为“精品课程”网页）。

编　者

2009 年 10 月

# 目　录

# 第一篇　基础实验

# 实验一　顺序表实验

**实验目的**：理解顺序表的逻辑结构和存储结构，熟练掌握顺序表的相关操作。

## 1. 问题描述

顺序表是指采用顺序存储结构的线性表，它利用内存中的一片起始位置确定的连续存储区域来存放表中的所有元素。可以根据需要对表中的任何数据元素进行访问，元素的插入、删除可以在表中的任何位置进行。

## 2. 数据结构设计

由于C语言中一维数组（即向量）也是采用顺序存储表示，因此可用一维数组 data[MAXSIZE]来描述顺序表，其中 MAXSIZE 是一个预先设定的常数（如 100），至于顺序表的长度（即线性表中元素的数目）可用一个整型变量 length 来表示，元素类型假定为 ElemType（ElemType 可以是任何相应的数据类型如 int，char 等），用结构类型来定义顺序表的类型。

```
#define MAXSIZE 100  //MAXSIZE 为线性表可能的最大长度
#define ERROR   -1
typedef struct
{
    ElemType data[MAXSIZE];
    int length;    //length 为线性表的长度
} SqList;          //线性表定义
```

## 3. 功能（函数）说明

顺序表的基本操作：

```
InitList(SqList &L)  //初始化操作，将线性表 L 置空
CreatSqlist(SqList &L,int n)  //建立一个顺序存储的线性表
Output(SqList L)  //输出顺序表 L
IsEmpty(SqList L)  //判断表是否为空.如果 L 是空表，返回
GetElem(SqList L,int i)  //取表 L 中第 i 元素
LocateElem(SqList L,ElemType x)
  //定位函数.返回 L 中第 i 个与 x 相等的数据元素的位置(从算起).否则返回值为-1
Insert(SqList &L,ElemType x,int i)
  //在线性表 L 中第 i(0≤i≤L.length)个数据元素之前插入一个数据元素 x
Delete(SqList &L,int i)
  //删除线性表 L 中第 i(0≤I<L.length)个数据元素
Clear(SqList &L)  // 清空线性表 L
```

```
MergeList(SqList la,SqList lb,SqList &lc)
  //合并有序表 la 和 lb 到表 lc 中,使得 lc 依然有序
```

## 4. 界面设计

指导用户按照正确的格式输入数据,并且使界面友好。

## 5. 编码实现

```
#define MAXSIZE 100    //MAXSIZE 为线性表可能的最大长度
#include<stdio.h>
#include<iostream.h>
typedef int  ElemType;
typedef struct
{
    ElemType data[MAXSIZE];
    int length;     // length 为线性表的长度
} SqList;           //线性表定义

void InitList(SqList &L)  //初始化操作,将线性表 L 置空
{
    L.length=0;
}

void CreatSqlist(SqList &L,int n)       //建立一个顺序存储的线性表
{
     int i;
     for(i=0;i<n;i++)
     scanf("%d",&L.data[i]);
     L.length=n;
     fflush(stdin);       //清除一个流
}

void Output(SqList L)     //输出顺序表 L
{
     int i;
     for(i=0;i<L.length;i++)
     printf("%5d",L.data[i]);     //每个数据占 5 列
     printf("\n");
}

int IsEmpty(SqList L)
  //判断表是否为空.如果 L 是空表,返回 1,否则返回 0
```

```
{
    if(L.length==0) return 1;
    else return 0;
}

int GetElem(SqList L,int i)        //取表中第 i 元素
{
    if(i<0 || i>=L.length)  return -9999;
    else return L.data[i];     //C 语言中数组的下标从"0"开始
}

int LocateElem(SqList L,ElemType x)
//定位函数.返回 L 中第 1 个与 x 相等的数据元素的位置(从 0 算起),否则返回值为 0
{
    int k=0;
    while(k<L.length && L.data[k]!=x)
    k++;
    if(k<L.length) return k;
    else return -1;
}

int Insert(SqList &L,ElemType x,int i)
//在线性表 L 中第 i(0≤i≤L.length)个数据元素之前插入一个数据元素 x
{
    int k;
    if(i<0 || i>L.length || L.length==MAXSIZE)
        return 0;
    else
    {   for(k=L.length;k>=i;k--)
        L.data[k]=L.data[k-1];
        L.data[i]=x;
        L.length=L.length+1;
    }
    return 1;
}

int Delete(SqList &L,int i)
    //删除线性表 L 中第 i(0≤I<L.length)个数据元素
{
    int k;
    if(i<0 || i>=L.length)  //下标越界
    return 0;
```

```
    else  //移动后面的元素
    {     for(k=i;k<L.length;k++)
          L.data[k]=L.data[k+1];
          L.length--;
    }
    return 1;
}

void Clear(SqList &L)  //清空线性表 L
{
    InitList(L);
}

void MergeList(SqList la,SqList lb,SqList &lc)
//合并有序表 la 和 lb 到表 lc 中,使得表 lc 依然有序
{
    int i,j,k;
    i=j=k=0;
    while(i<la.length && j<lb.length)  //la 和 lb 均不空
    if(la.data[i]<lb.data[j])
    lc.data [k++]=la.data[i++];
    else if(la.data[i] >lb.data[j])
    lc.data[k++]=lb.data[j++];
    else  //la 和 lb 中的当前元素相等时,同时移动指针
    {
         lc.data[k++]=lb.data[j++];i++;
    }
    while(i<la.length)  //la 非空,lb 已空
    lc.data[k++]=la.data[i++];
    while(j<lb.length)  //la 已空,lb 非空
    lc.data[k++]=la.data[j++];
    lc.length=k;
}

void output()
{
    int i;
    for(i=0;i<10;i++)
    printf(" ");
    for(i=0;i<32;i++)
    printf("*");
    printf("\n");
```

```
}

void mainpp()
{
    int i;
    output();
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("1.建立一个顺序表");
    for(i=0;i<10;i++)  printf(" ");printf("*");printf("\n");
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("2.输出一个顺序表");
    for(i=0;i<10;i++)  printf(" ");printf("*");printf("\n");
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("3.在顺序表中查找");
    for(i=0;i<10;i++)  printf(" ");printf("*");printf("\n");
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("4.向顺序表中插入一个元素");
    for(i=0;i<2;i++)  printf(" ");printf("*");printf("\n");
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("5.删除顺序表中的一个元素");
    for(i=0;i<2;i++)  printf(" ");printf("*");printf("\n");
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("6.从顺序表中取出一个元素");
    for(i=0;i<2;i++)  printf(" ");printf("*");printf("\n");
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("7.将两个顺序表合并");
    for(i=0;i<8;i++)  printf(" ");printf("*");printf("\n");
    for(i=0;i<10;i++)  printf(" ");printf("*    ");
    printf("0.退            出");
    for(i=0;i<8;i++)  printf(" ");printf("*");printf("\n");
    output();
}

void main()    //主函数
{
    int n,i,k=1,m,x;
    SqList l,la,lc;
    InitList(l);
    mainpp();
    while(k)
     {    printf("请选择 0--7 :    ");
          scanf("%d",&m);
```

```
        getchar();
    switch(m)
    {
        case 0: return;
        case 1: {       printf("输入元素值,构建顺序表:\n");
                        printf("请输入顺序表元素的个数:  ");
                        scanf("%d",&n);
                        CreatSqlist(l,n);  //建立长度为 n 的顺序表 l
                        Output(l);  //输出表 l
                        break;}
        case 2: Output(l);printf("\n");break;
        case 3: {       printf("请输入要查找的元素值:  ");
                        scanf("%d",&x);
                        k=LocateElem(l,x);
                        printf("要查找的元素的定位:%d\n",k);
                        printf("\n");
                        break;}
        case 4: {       printf("输入要插入元素的位置及其值:  ");
                        fflush(stdin);     //清除一个流
                        scanf("%d",&i);
                        scanf("%d",&x);
                        Insert(l,x,i);
                        Output(l);  //输出插入元素后的表 l
                        printf("\n");
                        break;}
        case 5: {       printf("输入要删除元素的位置:  ");
                        fflush(stdin);     //清除一个流
                        scanf("%d",&i);       Dlete(l,i);
                        Output(l);  //输出删除元素后的表 l
                        break;}
        case 6: {       printf("请输入要取出的元素的序号:  ");
                        fflush(stdin);     //清除一个流
                        scanf("%d",&i);
                        k=GetElem(l,i);       //取表中第 i 元素
                        printf("取出的第%d 个元素为:%d\n",i,k);
                        break;}
        case 7: {       InitList(la);
                        printf("请输入第 2 个顺序表元素的个数:  ");
                        scanf("%d",&m);
                        CreatSqlist(la,m); //建立长度为 m 的顺序表 l
                        Output(la);  //输出表 la
                        MergeList(l,la,lc);
```

```
                printf("输出合并后的顺序表中的元素:\n  ");
                Output(lc);  //输出表 lc
                break;}
            default :return;
        }
            printf("继续运行吗 Y(1)/N(0):  ");  scanf("%d",&k);
            if(!k) return;
    }
}
```

## 6. 运行测试

建立顺序表，按提示输入相关数据，调用相关功能函数，测试各功能函数。

# 实验二　链式存储实验

**实验目的**：理解链表的逻辑结构和存储结构，熟练掌握链表的相关操作。

## 1. 问题描述

链表是用一组任意的存储单元来依次存储线性表中的各个数据元素，这些存储单元可以是连续的，也可以是不连续的。用链接存储结构表示线性表的一个元素时至少要有两部分信息：一是这个数据元素的值，二是这个数据元素的直接后继的存储地址。这两部分信息一起组成了链表的一个结点。数据域用来存放数据元素的值；指针域（又称链域）用来存放该数据元素的直接后继结点的地址。链表正是通过每个结点的指针域将线性表的 n 个结点按其逻辑次序链接成为一个整体。通常用箭头表示链域中的指针，于是单链表就可以直观地画成用箭头链接起来的结点序列，单链表中每个结点的存储地址存放在其直接前驱的指针域中，因此访问单链表的每一个结点必须从表头指针开始进行。对单链表的操作主要有：建立单链表、查找（按序号查找、按值查找）、插入一个结点、删除一个结点、求表长等。

## 2. 数据结构设计

单链表的结点结构如下：

```
typedef struct node
{  //单链表结点结构
   ElemType  data;  //ElemType 可以是任何相应的数据类型如 int,char 等
   structnode *next;
} LinkList;
```

## 3. 功能（函数）说明

链表的基本操作：

```
InitList1(LinkList *&head)
    //初始化不带头结点的链表头指针
AddHead1(LinkList *&head,ElemType x)
    //使用表首添加法，向头指针为 head 的链表中插入一个结点，其值为 x
InitList(LinkList *&head)
    //初始化带头结点的链表头指针
AddHead(LinkList *head,ElemType x)
    //使用表尾添加法，向头指针为 head 的链表中插入一个结点，其值为 x
CreatList(LinkList *head,int n)
    //生成含有 n 个结点的单链表
output(LinkList *head)  //输出单链表
```