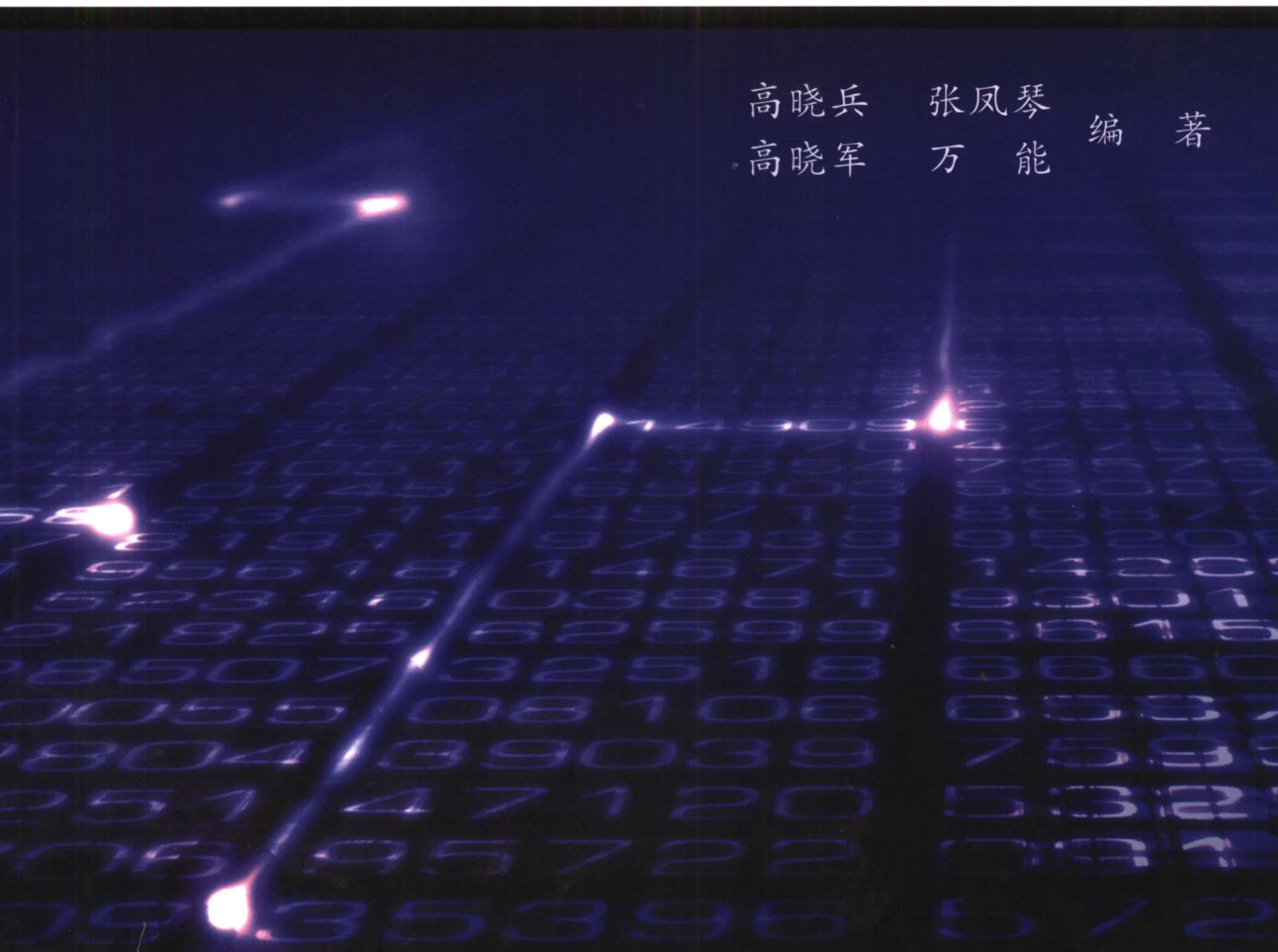
21世纪 高等学校电子信息类专业规划教材

数据结构实验教程

高晓兵 张凤琴
高晓军 万 能 编著



清华大学出版社
http://www.tup.tsinghua.edu.cn

北京交通大学出版社
http://press.bjtu.edu.cn

21 世纪高等学校电子信息类专业规划教材

数据结构实验教程

高晓兵　张凤琴　高晓军　万　能　编著

清华大学出版社
北京交通大学出版社
·北京·

内 容 简 介

《数据结构实验教程》是为了让学生能够尽快地掌握数据结构中的各种算法而编写的。本教材所写的算法具有程序结构清晰、可读性强、符合软件工程的规范要求等特点，所有的程序均在 VC 调试环境下运行通过，如果要运行程序，则仅需要编译一下便可。如果需要在 TURBO C 环境下运行，则仅需要将“//”注释修改一下便可。本书在数据结构的每个知识点上均给出了多个实验项目，且在每个实验项目中包括实验项目、任务分析、程序构思、源程序、测试数据、注意事项及思考问题等。在最后一章中给出了两个实际问题，着重分析了解决的思路、模块划分、重点难点等。本书共分 9 章，包括线性表、栈与队列、串、数组、树和二叉树、图、查找、排序和文件。

本书是清华大学出版社和北京交通大学出版社出版的《数据结构》教材（张凤琴主编）的配套实验教材，也可作为其他数据结构的实验教材及软件水平考试、计算机等级考试的上机指导、程序员编写算法的参考书。

图书在版编目（CIP）数据

数据结构实验教程/高晓兵等编著. —北京：清华大学出版社；北京交通大学出版社，2006.5
（21 世纪高等学校电子信息类专业规划教材）
ISBN 7－81082－782－0

Ⅰ. 数…　Ⅱ. 高…　Ⅲ. 数据结构－实验－高等学校－教材　Ⅳ. TP311.12－33

中国版本图书馆 CIP 数据核字（2006）第 048582 号

责任编辑：杨祎　　特邀编辑：吴炳林
出 版 者：清 华 大 学 出 版 社　　邮编：100084　　电话：010－62776969
　　北京交通大学出版社　　邮编：100044　　电话：010－51686414
印 刷 者：北京鑫海金澳胶印有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　　印张：11.5　　字数：277 千字
版　　次：2006 年 6 月第 1 版　　2006 年 6 月第 1 次印刷
书　　号：ISBN 7－81082－782－0/TP·276
印　　数：1～4 000 册　　定价：18.00 元

前　言

数据结构是计算机科学与技术及相关专业的基础课程，为了使学生能够尽快地掌握可读性好、执行速度快、占用空间少、可靠性高的程序的编写方法与技巧，从而达到在面对一个具体应用问题时，能够选择最佳的逻辑结构、存储结构及实现算法的目的，编写了这本数据结构的实验教材。本实验教材是清华大学出版社和北京交通大学出版社出版的《数据结构》教材（张凤琴主编）的配套实验教材。

本书实验项目安排合理，由浅入深，详略得当；在文字表达上力求简练清楚、概念清晰、通俗易懂；实验项目具体，操作性强，易于调试；描述的数据结构和算法结构清晰、可读性高、符合软件工程的规范。本书的所有实验项目使用 C ++ 语言进行描述，配有相关的解释，以帮助读者理解，并且所有的实验均在 VC 调试环境下运行通过的，如果要运行程序，则仅需要编译一下便可。如果需要在 TURBO C 环境下运行，则仅需要将“//”注释修改为“/ * … * /”便可。本书在数据结构的每个知识点上均给出了多个实验项目，且在每个实验项目中包括实验题目、任务分析、程序构思、源程序、测试数据、注意事项及思考问题等。整个实验项目的安排便于学生通过实验题目、任务分析和程序构思的理解，更容易看懂后边的源程序代码和测试数据，而注意事项提示学生本实验容易出错的地方，思考问题可以开阔学生的思路，最终起到举一反三的作用。最后一章中给出了两个实际问题，着重分析了解决的思路、模块划分、重点难点等，没有给出源程序，但学生通过前面实验项目的学习能够自己完成。

全书共分 9 章。主要内容如下：第 1 章是有关线性表的实验；第 2 章是有关栈与队列的实验；第 3 章是有关串的实验；第 4 章是有关数组的实验；第 5 章是有关树和二叉树的实验；第 6 章是有关图的实验；第 7 章是有关查找的实验；第 8 章是有关内部排序的实验；第 9 章是有关文件的实验。每一章均有知识点概述和小结，用于学生加深理解本章的内容，从而巩固知识和提高学习效率。本书前言后面附有实验教程说明，供需要自己解决实验思考题目的读者参考。

本书的第 1 章由张凤琴编写；第 2 ~ 5 章由高晓兵编写；第 6 ~ 8 章由高晓军编写；第 9 章由万能编写。张凤琴和高晓兵制定了编写大纲，最后由张凤琴对全文进行通审和定稿。

本书的整体框架由张凤琴副教授和高晓兵博士完成。在编写过程中，空军工程大学张水平教授、殷肖川副教授和西北工业大学的赵正文教授对本书的编写提出了宝贵的意见；研究生张青凤和李曼对本书的部分代码进行了调试，还得到了许多老师和研究生的支持。在此对他们的辛勤付出表示衷心的感谢。

由于作者水平有限，书中错误和疏漏之处在所难免，敬请读者指正，以便再版时纠正，我们编写人员在此表示深深的谢意。联系邮箱：Fengqin_zhang@126.com。

编　者

2006 年 4 月

实验教程使用说明

本实验教程中的源程序主要由四部分组成:库文件和预设宏定义、数据存储结构定义、函数定义和实现、主函数调用。对于需要读者自己解决的实验思考题目,参照如下所示的一般结构,就能顺利地编写数据结构程序。本书以 Visual C++6.0 为调试环境,程序行的注释有两种,一种是"/*…*/",另一种是以"//"开头。

```
//-----库文件和预设宏定义
#include <stdlib.h>
#include <stdio.h>
#define OK          1
#define ERROR       0
#define OVERFLOW -1
//定义数据存储结构
typedef struct
{
    int a;
    int *link;
}Name;
/*****------------------------------*****/
// 函数名:  函数名称
// 参数:    参数描述
// 返回值:  此函数返回值释义
// 功能:    描述此函数的功能作用
/*****------------------------------*****/
void Function1(Para1, Para2)
{
    /*注释*/
}//函数结束
int Function2(Para1, Para2)
{
    //注释
    return OK;
}//函数结束
//主程序-------------------------------------------
void main()
{
    //注释--调用函数 Function1
    Function1(p1, p2);
    /*
        注释
    */
}
```

目　录

第1章 线 性 表

本章的典型实验主要练习线性表的顺序、链式存储的基本操作和初步应用，分为线性表的顺序存储实验、线性表的单链表实验和线性表应用实验三类，共5个实验内容。

1.1 知识点概述

线性表是最基本最常用的数据结构，它有且仅有一个开始结点，该结点没有前驱且仅有一个后继；有且仅有一个终端结点，该结点没有后继且仅有一个前驱；其他所有结点都是内部结点，并且都有一个前驱和一个后继。一个线性表中的数据元素应具有相同的描述性质，即属于同一个数据对象。

线性表基本操作有初始化、判断表空、求表长、插入元素、删除元素等。

在实际应用中，必须将线性表中的数据存放在计算机中。常用的存储方式有两种：顺序存储和链式存储。顺序存储是用一块连续的地址存储单元依次存放线性表中的数据元素，使得数据元素的逻辑上的相邻关系与物理上的相邻关系一致。链式存储是用任意的存储单元存放线性表中的数据元素，这些单元可以分散在内存中的任一位置上，使得在表示数据结构时不能再用物理上相邻关系，而必须在存储每个元素的同时，也要存储元素之间的逻辑关系。这种存储单元只有在需要的时候才申请，不用事先分配；同样在使用完毕，应立即释放。

顺序存储的线性表又称顺序表，其特点是可以方便地存取表中任一元素；无需为表示元素间的逻辑关系而增加额外的存储空间；插入和删除运算时需移动大量元素，其效率较低；在长度变化较大的线性表预分配空间时，必须按最大空间分配，存储空间得不到充分利用；一旦分配好空间，表的容量难以扩充。

链式存储的线性表又称链表，其特点是查找表中任一元素时需从头结点的指针域开始逐步向后（前）查找；每个结点需要增加指针域；动态分配存储空间，存储空间得到了充分利用；容易插入和删除数据元素。

1.2 线性表的顺序存储实验

【实验1.1】 顺序表基本操作的设计与实现

实现顺序表的基本操作，包括顺序表的建立、查找、求长度、查找前驱、插入、删除、输出等函数。

第1步：任务分析。

完成顺序表的建立、查找、求长度、查找前驱、插入、删除、输出等函数功能，有助于更好的理解顺序表的概念和用法。上述这些函数都是线性表的基本操作，根据这些基本操作，可构成其他更复杂的操作。

第2步：程序构思。

顺序表的建立，首先要建立一块连续存储的空间。在Turbo C或C++中由于数组采用顺序存储，所以使用数组这种数据结构来描述线性表的顺序存储结构。顺序表的存储结构描述包含存储空间和实际长度两个重要的数据，这两个重要数据在实际实现时可以灵活变化，参见思考描述。

第3步：源程序。

```
//调试环境：  Visual C++ 6.0

//- - - - -库文件和预设定义
#include <stdlib.h>
#include <stdio.h>

#define OK              1
#define ERROR           0
#define  OVERFLOW       -1

#define List_INIT_SPACE     100     //存储空间初始分配量
#define List_INC_SPACE      10      //存储空间分配增量

typedef int ElemType;               //指定顺序表中数据类型

typedef  struct
{
   ElemType *elem;                  //存储空间基址
   int     length;                  //当前长度
   int     listsize; //当前分配的存储容量(以 sizeof(ElemType)为单位)
}Sq_List;

/* * * * - - - - - - - - - - - - - - - - - - - - - - - -   * * * * */
// 函数名:    Sq_ListInit (Sq_List &L)
// 参数:      (传入) SqList L,顺序表结构体 L,存储线性表相关信息(& 相当于传
              入 L 的地址)
// 返回值:    int 型,返回 1 表示创建成功,0 表示失败
// 功能:      初始化一个空顺序表
/* * * * - - - - - - - - - - - - - - - - - - - - - - - -   * * * * */
int Sq_ListInit ( Sq_List &L)
{
    //在内存中分配空间
    L.elem = ( ElemType * ) malloc (List_INIT_SPACE * sizeof ( ElemType));
    if (! L.elem)  exit (OVERFLOW);     //存储分配失败

    // 构造一个空的线性表 L  //
```

```
36          L.length=0;
37          L.listsize=List_INC_SPACE;          //初始存储容量
38          return  OK;
39      }//函数 Sq_ListInit 结束
40
41      /*****------------------------------*****/
42      // 函数名:    LocateElem(Sq_List L,ElemType e)
43      // 参数:      (传入) Sq_List L,顺序表
44      //            (传入) ElemType e,定位元素
45      // 返回值:    int 型,返回定位位置,0 表示失败
46      // 功能:      在顺序表中定位元素
47      /*****------------------------------*****/
48      int LocateElem(Sq_List L,int e)
49      {
50          int i=1;
51          //定义线性表指针
52          int *p=L.elem;
53          //查找元素 e
54          while(i<=L.length&&*p++!=e)
55              ++i;
56
57          if(i<=L.length)
58              return i;
59          else
60              return ERROR;
61      }
62      /*求顺序表的长度*/
63      int GetListLength(Sq_List L)
64      {
65          return L.length;
66      }
67
68      /*****------------------------------*****/
69      // 函数名:    Sq_ListInsert ( Sq_List &L, int i, ElemType e )
70      // 参数:      (传入) Sq_List &L     顺序表
71      //            (传入) int i 插入位置
72      //            (传入) ElemType e  插入元素
73      // 返回值:    1 表示成功,0 表示操作失败
74      // 功能:      在顺序表 L 中的第 i 个位置前插入新元素 e
75      // 备注:      i 的合法取值为 1≤i≤线性表长度+1
76      /*****------------------------------*****/
77      int Sq_ListInsert ( Sq_List &L, int i, ElemType e )
78      {
```

```
79      //判断位置是否合法
80            if ( i < 1 ||i > L.length +1)
81            {
82               printf("i 的值不合法!\n");
83              return 0;
84           }
85
86           //超出空间进行再分配
87           if ( L.length >=L.listsize )
88           {
89              int * newspace;
90      newspace = ( ElemType * ) realloc ( L.elem, (L.listsize + List_INC_SPACE)
        * sizeof ( ElemType ));
91              if (! newspace)  exit (OVERFLOW);          //存储分配失败
92              L.elem = newspace;                         //新基址
93              L.listsize + = List_INC_SPACE;             //增加存储容量
94           }
95
96           int *p, *q;                //定义指向线性表位置 i 和尾的指针
97           q = &(L.elem[i -1]);       //q 指针指向插入位置 i 的前一个
98           for(p = &(L.elem[L.length -1]);p >= q; --p)
99                 * (p +1) = * p;      //插入元素之后的元素右移
100          *q = e;                    //把元素 e 放在位置 i 处
101          ++L.length;                //线性表长度增 1
102          return  OK;
103     }//函数 Sq_ListInsert 结束
104
105     /* * * * - - - - - - - - - - - - - - - - - - - - - - - -   * * * * */
106     // 函数名:    Sq_ListDelete ( Sq_List &L, int i, ElemType &e )
107     // 参数:      (传入) Sq_List &L     顺序表
108     //            (传入) int i      删除位置
109     //            (传出) ElemType &e 删除元素
110     // 返回值:    1 表示成功,0 表示操作失败
111     // 功能:      在顺序线性表 L 中删除第 i 个元素,用 e 返回其值
112     // 备注:      i 的合法取值为 1≤i≤线性表长度
113     /* * * * - - - - - - - - - - - - - - - - - - - - - - - -   * * * * */
114     int Sq_ListDelete ( Sq_List &L, int i, ElemType &e )
115     {
116        //判断位置是否合法
117        if ( i < 1 ||i > L.length +1)
118        {
119           printf("i 的值不合法!\n");
120           return 0;
```

```
121         }
122
123         int *p,*q;                              //定义指向线性表位置 i 和尾的指针
124            p=& (L.elem[i-1]);                   //p 为被删除元素的位置
125            e=* p;                               //取删除元素的值
126            q=L.elem + L.length-1;               //q 指针指向线性表最后一个元素
127            for (++p; p<=q; ++p)
128                    * (p-1) = *p;                //被删除元素之后的元素左移
129            --L.length;                          //线性表长度减 1
130            return  OK;
131        }//函数 Sq_ListDelete 结束
132
133      //- - - - - - - - - - - -测试程序 - - - - - - - - - - - -
134        void main()
135        {
136            Sq_List R;//实参定义
137            int flag;
138            //为了判断调用成功与否,可以检查 flag 的值,或者见插入操作的调用方法
139            //  if(flag==1)  printf("成功");
140            //  else  printf("失败");
141            flag=Sq_ListInit(R);
142
143            //调用方法
144            int i=1;
145            int elem;
146            printf("the length of current list is %d\n", GetListLength(R));
147            printf("input the element:");
148            scanf("%d", &elem);
149            if( Sq_Listlnsert(R,i,elem) )    printf("successed!\n");
150            printf("the length of current list is %d\n", GetListLength(R));
151            //Sq_Listlnsert(R,i,elem);    //或者直接调用
152        }
```

第4步:测试数据。

运行结果:

```
the length of current list is 0
input the element:45
successed!
the length of current list is 1
```

注意:

从这个实验开始,程序使用了 malloc 函数,此函数向系统请求分配内存空间,其参数为申请的内存空间的大小,通常由求字节长度的 sizeof 函数计算得到空间大小。申请空间成

功时返回指向该空间起始地址的指针(由于 malloc 默认返回 void 类型指针,所以要强制类型转换),否则返回 NULL 指针。使用此函数,必须包含对应的头文件 stdlib. h。

思考题:

采用直观的数组定义来实现顺序表,规定数组中下标为 0 的元素存储顺序表的实际长度,这时顺序表存储的元素从下标 1 开始。具体定义举例如下:

```
#define MAX 100
int A[MAX+1];        /* 存储空间初始分配量为100,下标访问从1~100 */
A[0]=10;             /* 顺序表当前长度为10 */
```

试根据上述定义,完成顺序表的基本操作。

【实验 1.2】 顺序表基本操作应用实验

应用题目:

(1) 有一个表元素按值递增排列的顺序表,编写一个函数实现删除顺序表中多余的值相同元素;

(2) 有一个顺序表,编写一个在顺序表中查找最大和最小值元素的函数,并分析其时间复杂度 $T(n)$。

第 1 步:任务分析。

这两个题目涉及顺序表元素的删除、访问等,重点在于在顺序表的基础上领悟程序算法设计。

第 2 步:程序构思。

(1) 由于表中元素按照其值非递减排列,值相同的元素必为相邻的元素,故程序实现时,依次比较相邻的两个元素,若值相等,删除其中一个,否则继续向后查找。

(2) 依次扫描一遍表中所有元素,比较查找最大和最小值。

第 3 步:源程序。

```
1    /* * * * - - - - - - - - - - - - - - - - - - - - - - - -  * * * * */
2    // 函数名:    delete(List A, int n)
3    // 参数:      List A 顺序表, int n 顺序表长度
4    // 返回值:    无
5    // 功能:      删除顺序表中多余的值相同元素
6    // 备注:      顺序表用一维数组实现,故 List A 可以理解为 int A[N]
7    /* * * * - - - - - - - - - - - - - - - - - - - - - - - -  * * * * */
8    void delete(List A, int n)
9    {
10       int i=0, j;
11       while(i<=n-2)
12           if(A[i]!=A[i+1])i++;     /*元素值不等,继续查找*/
13           else;
14           {
15           for(j=(i+2);j<=n;j++)A[j-1]=A[j];  /*删除第 i+1 个元素*/
```

```
16                  n--;    /*表长度减1*/
17              }/* end if */
18      }
19
20      /*****------------------------*****/
21      // 函数名:    maxmin(List A, int n)
22      // 参数:      List A顺序表, int n顺序表长度
23      // 返回值:    无
24      // 功能:      查找最大和最小值元素
25      // 备注:      顺序表用一维数组实现,故List A可以理解为int A[N]
26      /*****------------------------*****/
27      void maxmin(List A, int n)
28      {
29          int max,min,i;
30          max=A[0];min=A[0];
31          for(i=1;i<n;i++)
32          {
33              if(A[i]>max)    max=A[i];
34              else if(A[i]<min)    min=A[i];
35          }
36          printf("max=%d, min=%d\n", max, min);
37      }
```

第4步：测试。

请读者自行完成main( )函数。

思考题：

第二个题目函数关于时间复杂度$T(n)$的讨论，请从下面两种特殊情况分别考虑程序的比较次数。

(1) 最坏情况(顺序表递减排列)下的比较次数；

(2) 最好情况(顺序表递增排列)下的比较次数。

1.3　线性表的链表实验

【实验1.3】　单链表的的设计与实现

实现单链表的基本操作，包括链表的建立与释放、查找、求长度、查找后继、插入、删除、输出等函数。

第1步：任务分析。

单链表的建立、查找、求长度、查找前驱、插入、删除、输出等是链表的基本操作，正确的实现它们，需要首先从存储形式上理解链表。单链表的存储结构描述包含数据域(存储数据元素值)和指针域(存储地址，指向链表下一个节点的指针)两部分，这两部分构成一个结点，具体用C语言的结构体来描述。链表的操作主要是要清楚指针的变化情况，明确头指

针、当前指针的指向，为了更好的理解和掌握，可以以图示的形式进行。请参见教材相关部分。

第2步：程序构思。

1. 单链表的建立

(1) 先建立头节点 head，将头节点的指针域置为空。

(2) 新建一个节点 p，把此新节点链接到单链表的尾端(p -> next 设为空)或者始端。

① p -> next 指向头节点指向的下一个节点 head -> next。

② head -> next 指向 p。

2. 单链表的插入

(1) 新建一个节点 p，指定插入位置。

(2) 从单链表头开始查找节点位置。

① 找到该位置，则 p -> next 指向当前位置的下一个节点，当前位置节点指向新建节点 p。

② 没有找到该位置，插入操作失败。

3. 单链表的删除

(1) 指定删除位置。

(2) 从单链表头开始查找节点位置。

① 找到该位置，则删除该位置的节点。

② 没有找到该位置，删除操作失败。

第3步：源程序。

```
1        //调试环境:Visual C++ 6.0
2
3        //-----库文件和预设定义
4        #include <stdlib.h>
5        #include <stdio.h>
6        #define NULL 0
7
8        typedef int ElemType;//指定单链表中数据类型
9
10       //单链表存储结构定义
11       typedef struct LNode
12       {    ElemType data;          //数据域
13            struct LNode *next;     //指针域
14       }LNode, *LinkList;
15
16       /*****------------------------------*****/
17       // 单链表建立方法一(函数用返回值得到表头指针)
18       // 函数名:    CreateOne(int n)
19       // 参数:      (传入)int n,传入线性表结点数量
20       // 作用:      建立一个空线性表
```

```
21      // 返回值:    LNode * 型返回结构体指针,即得到建立的线性表头指针
22      /* * * * - - - - - - - - - - - - - - - - - - - - - - - -  * * * * */
23      LNode * CreateOne(int n)
24      {
25        int i;
26        LNode * head;/ * 定义头结点指针 * /
27        LNode *p;/ * 定义新结点指针 * /
28
29        / * 建立带头结点的线性链表 * /
30        head = (LNode * )malloc(sizeof(LNode));
31        head -> next = NULL;
32
33        printf("Please input the data for LinkList Nodes:\n");
34        for(i = n;i > 0; -- i)
35        {
36            p = (LNode * )malloc(sizeof(LNode));/ * 为新结点申请空间,即创建一新
              结点 * /
37            scanf("%d",&p -> data); / * 新结点赋值 * /
38            / * 新结点插入到表头 * /
39            p -> next = head -> next;
40            head -> next = p;
41        }
42        return head;/ * 返回头结点指针,即可以得到单链表的地址 * /
43      }
44
45      /* * * * - - - - - - - - - - - - - - - - - - - - - -  * * * * */
46      // 单链表建立方法二(函数无返回值)
47      // 函数名:    CreateTwo(LNode * head, int n)
48      // 参数:      (传入)LNode * head 传入一个链表指针
49      //            (传入)int n,传入线性表结点数量
50      // 作用:      建立一个空线性表
51      // 返回值:    无
52      /* * * * - - - - - - - - - - - - - - - - - - - - - -  * * * * */
53      void CreateTwo(LinkList &head, int n)
54      {
55        int i;
56        LNode *p;/ * 定义新结点指针 * /
57
58        / * 建立带头结点的线性链表 * /
59        head = (LinkList)malloc(sizeof(LNode));
60        head -> next = NULL;
61
62        printf("Please input the data for LinkList Nodes:\n");
```