21世纪高等学校规划教材

SHUJU JIEGOU SHIYAN ZHIDAO

数据结构实验指导

(C语言版)

杨晓波 主编
王 弘 王聪华 胡 永 副主编

中国电力出版社
http://jc.cepp.com.cn

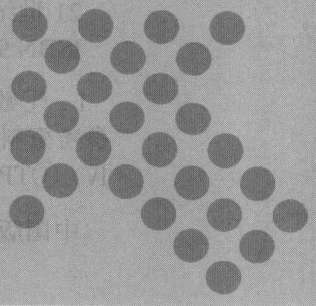
21世纪高等学校规划教材

SHUJU JIEGOU SHIYAN ZHIDAO

数据结构实验指导

(C语言版)

主　编　杨晓波
副主编　王　弘　王聪华　胡　永
编　写　张晓煜　雷　萌
主　审　杨兴运



中国电力出版社
http://jc.cepp.com.cn

内 容 提 要

本书为 21 世纪高等学校规划教材。本书是《数据结构（C 语言版）》一书的配套上机实验指导书，两本书章次一一对应。全书共分三个部分：第一部分上机实验；第二部分习题解析；第三部分模拟试题及参考答案。本书力求语言通俗、算法表达精练、易读易懂、注重应用实践，注意训练学生的实际应用和上机动手能力。

本书适用于应用型本科计算机及相关专业的教学，也可供自学人员及工程技术人员学习参考。

图书在版编目（CIP）数据

数据结构实验指导：C 语言版/杨晓波主编. —北京：中国电力出版社，2010.1

21 世纪高等学校规划教材

ISBN 978-7-5083-9851-8

Ⅰ. ①数… Ⅱ. ①杨… Ⅲ. ①数据结构－高等学校－教学参考资料 ②C 语言－程序设计－高等学校－教学参考资料 Ⅳ. ①TP311.12 ②TP312

中国版本图书馆 CIP 数据核字（2010）第 000403 号

中国电力出版社出版、发行

（北京三里河路 6 号 100044 http://jc.cepp.com.cn）

汇鑫印务有限公司印刷

各地新华书店经售

2010 年 2 月第一版 2010 年 2 月北京第一次印刷

787 毫米×1092 毫米 16 开本 12.75 印张 307 千字

印数 0001—3000 册 定价 **20.40** 元

前　言

本书是《数据结构（C语言版）》的配套上机实验指导书。

全书共分三个部分：第一部分为上机实验；第二部分为习题解析；第三部分为模拟试题及参考答案。第一部分和第二部分的章节与教材一一对应，第1章绪论，第2章线性表，第3章栈和队列，第4章串、数组和广义表，第5章树和二叉树，第6章图，第7章查找，第8章内部排序。上机实验中给出了各种数据结构的基本运算并配有实际应用实例。习题解析中习题种类丰富，并给出解答。另外书后给出了两个附录，附录A较系统地给出在Visual C++ 6.0环境下编写C语言程序所需要的基本知识。附录B为实验报告的要求。

书中所有程序都在Visual C++ 6.0环境下调试通过。

本书在编排上注意完整性和独立性，自成一体，可单独作为上机实践指导和习题解答使用。

本书由杨兴运主审。本书在编写过程中，还参考了相关的实验指导书及教材，在此向这些作者表示感谢。

由于时间仓促和作者水平有限，书中存在的疏漏和谬误，请读者不吝赐教。

编　者

2009年10月

目　录

第二部分　课后习题及解析

第三部分　模拟试题及参考答案

第一部分　上　机　实　验

第1章　绪　　论

一、知识要点回顾

1. 数据结构的概念

数据结构是研究非数值计算的程序设计问题中计算机的操作对象以及它们之间的关系和运算等的专门学科。关系中仅仅考虑数据元素之间的逻辑关系，称为逻辑结构，逻辑结构分为：集合、线性结构、树结构和图结构；数据元素在计算机内的实际存储，称为物理结构或存储结构，基本的有顺序存储和链式存储。

2. C语言语法结构要点

（1）符号常量定义。

```
# define TRUE  1
# define FALSE  0
```

（2）类型别名定义。

用 typedef 为一个数据类型定义新的名字。如：

```
typedef  int  Bool;
typedef  int  Status ;
```

（3）运算符。

算术运算符：+、-、*、/、%（取余）；

比较运算符：== 、! =、<、>、<= 、>=；

逻辑运算符：||（或）、&&（与）、!（非）。

（4）结构定义。

1）用 typedef 定义结构类型。

```
typedef struct 结构类型名
{结构成员列表
}结构类型名表;
```

各结构类型名之间用逗号分隔。

2）定义结构类型的同时定义结构变量。

```
struct 结构类型名
{结构成员列表
}结构变量名表;
```

各结构变量名之间用逗号分隔。

（5）自定义函数。

```
<函数类型><函数名> (<参数表>){
```

```
 // 算法说明
< 局部变量说明>;
<语句序列>;
} // <函数名>
```

（6）基本语句。

1）计算赋值语句。

```
变量名=表达式;                                    // 单个变量赋值
变量名 1 =变量名 2 = ... =变量名 n =表达式;       // 串联赋值
结构名=结构名;                                    // 结构体变量整体赋值
结构名= {值 1 , 值 2, ...,值 n }
数组名[ ] = 表达式;                               // 数组初始化
数组名[ low ..high] = 数组名[ low ..high] ;       // 数组整体赋值
变量名←→变量名;                                   // 交换赋值
变量名=条件表达式? 表达式 1: 表达式 2             // 条件赋值
```

2）I/O 语句。

```
scanf ( <"格式描述"> , &变量名 1, &变量名 2 ,… , &变量名 k )     // 输入语句
printf ( <"格式描述"> , 表达式 1, 表达式 2 ,… , 表达式 n )      // 输出语句
getchar ( ) ;                                                  // 字符输入
putchar ( ) ;                                                  // 字符输出
```

3）注释语句。

```
// 注释内容                                                    // 单行注释
/* 注释内容                                                   */ 多行注释
```

4）条件选择语句。

```
单选择语句
    if (条件表达式)  语句序列 S ;
双选择语句
    if (条件表达式)  语句序列 S1;
      else 语句序列 S2 ;
多选择语句(开关语句)
    switch (表达式) {
      case <值 1 > :语句序列 1 ; break ;
      case <值 2 > :语句序列 2 ; break ;
      …
      case <值 n > :语句序列 n ; break ;
      default:语句序列 n+1 ;
     } // end  switch
```

5）循环语句。

```
计数循环语句
   for (循环变量=初值;循环变量<=终值;++/--循环变量++ /--)
     {语句序列; }
当循环语句
   while (条件表达式)
```

```
    {语句序列; }
重复循环语句
   do {
语句序列 ;
} while (条件表达式)
```

6）其他控制语句。

```
return <表达式> ;          // 返回语句
break ;                    // case或循环终止语句
exit (代码) ;              // 异常结束语句
```

二、上机实验

实验目的

（1）熟悉和复习C语言及Visual C++ 6.0编程环境。

（2）掌握Visual C++ 6.0环境下主要针对C语言的Win32控制台应用程序编程。

（3）掌握C语言函数的编写、函数的调用、输入和输出。

（4）掌握一维数组的使用，指针及引用类型的使用。

（5）掌握结构体及结构体指针变量的使用。

实验要求

（1）启动控制台程序，创建一个C语言主程序。

（2）加入包含文件#include “stdio.h”，利用scanf()和printf()函数输入、输出。

（3）加入包含文件#include “malloc.h”，调用函数malloc()给结构指针分配空间。

实验题目及解析

实验1.1　编写一主程序分别调用整数加法和乘法函数

考核内容：

（1）分别写出整数加法Add和乘法Mul的两个函数；

（2）主程序中以对话形式输入两个整数，分别调用整数加法和乘法函数，并用printf()输出结果。

参考程序：

```
#include "stdio.h"
int Add(int x,int y)
{// 求x,y的和
     return (x+y);
}
long Mul(int x,int y)
{// 求x,y的积
```

```
    return (x*y);
}
int main(int argc, char* argv[])
{
    int x,y,s;long m;
    printf("请输入两个整数 x,y:");
    scanf("%d %d",&x,&y);          // 用空格分开输入 x,y 两个整数
    s=Add(x,y);                    // 调用加法函数
    m=Mul(x,y);                    // 调用乘法函数
    printf("x+y=%d\n",s);
    printf("x*y=%ld\n",m);
    return 0;
}
```

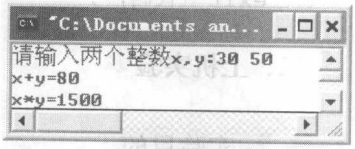


图 1-1 实验 1.1 程序运行结果

程序分析：根据屏幕提示，输入两个整数（注意用空格分开），然后按回车键，操作结果如图 1-1 所示。

实验 1.2 求最大、最小值

考核内容：

（1）编写一个求 *n* 个整数最大值和最小值的函数 GetMax_Min()；

（2）在函数 GetMax_Min()中，设计整型数组形式参数传入 *n* 个整数；

（3）设计参数返回最大值和最小值。

参考程序：

```
#include "stdio.h"
#include "stdlib.h"
void GetMax_Min(int a[],int n,int &Max,int &Min)
{// 求 n 个整数的最大值和最小值，并分别存入 Max 和 Min
    int i;
    Max=a[0];Min=a[0];             // 给最大值和最小值赋初值
    for(i=1;i<n;i++)
    {   if(a[i]>Max)  Max=a[i];    // 如果第 i 个值大于最大值第 i 个值为最大值
        if(a[i]<Min)  Min=a[i];    // 如果第 i 个值小于最小值第 i 个值为最小值
    }
}
main()
{   int i,Max,Min;int num[10];
    for(i=0;i<10;i++)
    {  num[i]=rand();// 随机生成 10 个整数
       printf("%d\n",num[i]); }
    GetMax_Min(num,10,Max,Min);
   printf("Max=%d\n",Max);
   printf("Min=%d\n",Min);
}
```

程序分析：函数 GetMax_Min()中的两个形式参数 int &Max,int &Min 是 Visual C++中的引用类型，它的好处在于调用函数时的实际参数可以是普通变量，实际参数传递的是变量地址，函数内部可以将运算结果存储到该地址的存储区，利用变量地址将运算结果返回给主程序。引用类型很有用处，在 Visual C++中也比较实用，希望读者通过 Internet 深入学习，能熟练运用引用类型。为了能将运算结果返回给主程序，函数 GetMax_Min()中的形式参数 Max 和 Min 也可以定义成指针变量 int *Max,int *Min，此时，主函数调用时，实际参数应采用取变量地址方式，如调用函数 GetMax_Min() 可以改成 GetMax_Min(num,10,&Max, &Min)的形式。

主程序中利用了随机函数 rand()随机生成 10 个整数，应在程序开始处加入包含文件 #include "stdlib.h"，程序才能识别函数 rand()。程序运行结果如图 1-2 所示。

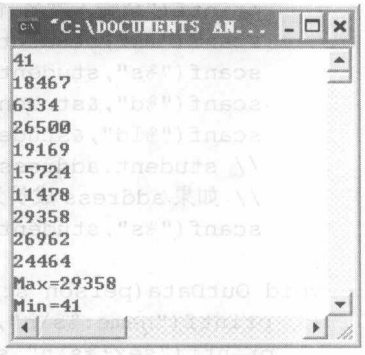


图 1-2 实验 1.2 程序运行结果

实验 1.3 结构体的应用

考核内容：

（1）已知一个人的基本信息，包括姓名（name）、性别（sex）、年龄（age）、联系电话（telephone）和地址（address），用结构体定义基本信息，通过函数给结构体成员赋值，并输出结构体成员的值；

（2）定义结构体 person，含有姓名（name），性别（sex），年龄（age），联系电话（telephone）和地址（address）成员；

（3）设计输入、输出函数，参数为结构体指针变量，输入函数内用 scanf()输入结构体各成员，输出函数内用 printf()输出结构体各成员。

参考程序：

```
#include "stdio.h"
#include "stdlib.h"
typedef struct
{    char name[8];
     char sex[2];
     int age;
     long telephone;
     char address[20];                  // char *address;
}person;                                // 用 typedef 定义结构体类型为 person
void GetData(person *student)           // 形式参数为结构体指针变量
{// 此函数同时说明对结构体指针成员的引用方法
    printf("请输入姓名,性别,年龄,电话号码,籍贯:\n");
    scanf("%s",student->name);
    scanf("%s",student->sex);
    scanf("%d",&student->age);
    scanf("%ld",&student->telephone);
    // student->address=(char *)malloc(sizeof(char)*20);
    // 如果 address 成员为字符指针时,必须为其申请空间,才能用 scanf()输入
```

```
    scanf("%s",student->address);
}
void Get_Data(person &student)                    // 形式参数为结构引用类型
{// 此函数同时说明对结构体变量成员的引用方法
    printf("请输入姓名,性别,年龄,电话号码,籍贯:\n");
    scanf("%s",student.name);
    scanf("%s",student.sex);
    scanf("%d",&student.age);
    scanf("%ld",&student.telephone);
    // student.address=(char *)malloc(sizeof(char)*20);
    // 如果 address 成员为字符指针时,必须为其申请空间,才能用 scanf()输入
    scanf("%s",student.address);
}
void OutData(person student)
{   printf("name:%s\n",student.name);
    printf("sex:%s\n",student.sex);
    printf("age:%d\n",student.age);
    printf("telephone:%ld\n",student.telephone);
    printf("address:%s\n",student.address);
}
main()
{   person student;// person *student;
    // student=(person*)malloc(sizeof(person));//按结构体指针传递参数时,
    // GetData(student);                        //必须为指针变量申请存储地址
    Get_Data(student);                          //结构体变量为实际参数时的调用
    OutData(student);//OutData(*student);       //结构体指针作为实际参数时的调用
}
```

程序分析：参考程序给出了两个输入函数 void Get_Data(person &student)和 void GetData (person *student)，它们的主要区别在于形式参数，一个是结构体引用类型，另一个是指向结构体的指针。在主程序中给出了按引用类型传递参数的实例，实际参数是按普通结构体变量来定义的，而主程序中加注释的语句是结构体指针作为实际参数的调用实例，实际参数是按结构体指针变量来定义的，特别要强调的是将结构体指针作为实际参数时，必须用 malloc()函数为结构体指针变量分配地址，否则，程序编译虽然通过，但运行输入时要出错误。定义的结构体地址成员 address 也可以定义成字符类型的指针，这样的好处是可以不计地址成员的长度，但在输入函数中，利用 scanf()输入之前，必须用 malloc()函数为其分配地址。程序中的注释也说明了这一点。主程序在执行时，每输入一行按一次回车键，运行结果如图 1-3 所示。

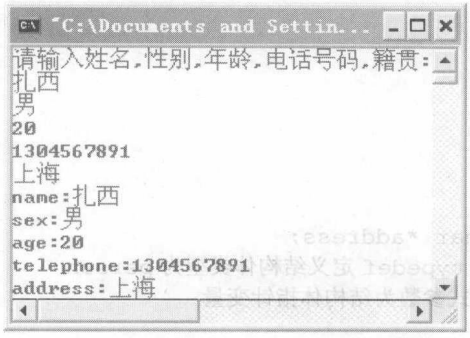


图 1-3 实验 1.3 程序运行结果

第2章　线　性　表

线性表是一种最基本、最常用的数据结构，它有两种存储结构——顺序表和链表。本章主要介绍线性表的顺序存储和链式存储两种存储结构，以及在这两种存储结构上进行的建立、插入、删除、查找等基本运算的实现。

一、知识要点回顾

1. 顺序表

顺序表是由地址连续的向量实现的，便于实现随机访问。

顺序表进行插入和删除运算时，平均需要移动表中大约一半的数据元素，容量难以扩充。

2. 单链表（线性链表）

单链表存储结构不需要使用地址连续的存储单元来实现，相关存储单元既可以是连续的，也可以是非连续的，通过指针建立数据元素之间的逻辑关系，操作比较灵活方便，但不能随机存取。

3. 双链表

在单链表的每个结点中增加一个指针域指向它的前驱结点，这样该链表中的每个结点都具有两个指针域，其中一个指针域指向直接后继，另一个指针域指向直接前驱，将该链表称为双链表。双链表查找每个结点的前驱和后继都很方便。

二、上机实验

实验目的

（1）熟悉将算法转换为程序代码的过程。

（2）了解顺序表的逻辑结构特性，熟练掌握顺序表存储结构的C语言描述方法。

（3）熟练掌握顺序表的基本运算：查找、插入、删除等，掌握顺序表的随机存取特性。

（4）了解线性表的链式存储结构及其顺序存取特性，熟练掌握线性表的链式存储结构的C语言描述方法。

（5）熟练掌握线性链表（单链表）的基本运算：查找、插入、删除等，能在实际应用中灵活选择适当的链表结构。

（6）掌握使用链表表示特定形式数据的方法，并能编写出相关运算的算法。

实验要求

（1）熟悉顺序表的插入、删除和查找。

（2）熟悉单链表的插入、删除和查找。

（3）熟悉双链表的插入、删除和查找。

实验题目及解析

实验 2.1 实现顺序表各种基本运算

考核内容：

（1）以顺序表作为存储结构；

（2）实现顺序表上的数据元素的插入运算；

（3）实现顺序表上的数据元素的删除运算；

（4）实现顺序表上的数据元素的查找运算。

参考程序：

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#define MaxSize 20
typedef int ElemType;
typedef struct SeqList
{// 线性表顺序存储结构定义
    ElemType elem[MaxSize];                    //存放顺序表数据元素
    int length;                                //当前长度
}SeqList;                                      //线性表顺序存储结构类型名
int Init_SeqList(SeqList &L)                   //顺序表初始化
{   L.length=0;     return 1;  }
int Locate_SeqList(SeqList &L,int x)           //在顺序存储结构的线性表中查找某个元素
{   int i=0;                                   //i 为扫描指示器,赋初值为 0
    while(i<L.length&&L.elem[i]!=x)            //顺序扫描直至找到值等于给定值的元素
                                               //或扫描至表尾
      i++;
    if(i>=L.length) {printf("顺序表中不存在该元素!\n");return 0; }
                                               //不存在值等于给定值的元素
    else    return i+1;                        //找到值等于给定值的元素
}
int Insert_SeqList(SeqList &L,int i,int x)
{// 在顺序存储结构的线性表的第 i 个数据元素之前插入新元素 x，1≤i≤n+1
    int j;
    if(L.length>=MaxSize)                      //L.length 是最后一个元素的位置
    {  printf("顺序表已满，无法进行插入操作!");return 0;    }
    if(i<=0||i>L.length+1)
    {  printf("插入位置不正确!");  return 0;  }//L.length 是最后一个元素的位置
    for(j=L.length-1;j>=i-1;j--)               //最后一个元素的下标是 length-1,第
                                               //i 个元素位置下标是 i-1
    {   L.elem[j+1]=L.elem[j]; }               //元素右移
    L.elem[i-1]=x;L.length++;                  //完成插入,表长增 1
    return 1;
}
int Delete_SeqList(SeqList &L,int i)
{// 在顺序存储结构的线性表中删除第 i 个元素,1≤i<L.length
```

```
    int j;
    if((i<1)||(i>L.length))
    {  printf("删除位置不正确!");return 0;  }
    for(j=i;j<L.length;j++)       L.elem[j-1]=L.elem[j];     //元素左移
    L.length--;                                              //表长减1
    return 1;
}
int Display_SeqList(SeqList L)
{// 顺序表的显示
    int i;
    for(i=0;i<L.length;i++)  printf("%d ",L.elem[i]);
    return 1;
}
void main()
{     SeqList L;
      ElemType e,x;
      int i=1,k,j;
      Init_SeqList(L);
      printf("初始化\n建立顺序表如下:\n");
      Insert_SeqList(L,1,1);
      Insert_SeqList(L,2,2);Insert_SeqList(L,3,3);
      Insert_SeqList(L,4,4);Display_SeqList(L);printf("\n");
      while(i<=3)
      {    printf("\n          主菜单        \n");
           printf("     1   查找指定元素    \n");
           printf("     2   插入元素到指定位置    \n");
           printf("     3   删除某一指定位置元素        \n");
           printf("     0   结束程序    \n");
           printf("-----------------------------------------\n");
           printf("请输入您选择的菜单号<1, 2, 3, 0>: ");
           scanf("%d",&i);
           switch(i)
           { case 1 :
               printf("请输入查找元素: "); scanf("%d",&x);
               j=Locate_SeqList(L,x);
               if(j!=0) printf("指定元素位置=%d\n",j);
               break;
             case 2:
               printf("请输入插入元素位置: ");scanf("%d",&k);
               printf("请输入插入元素值: ");    scanf("%d",&x);
               j=Insert_SeqList(L,k,x);
               if(j!=0)
               {printf("插入后顺序表如下所示\n");Display_SeqList(L);printf ("\n");}
               break;
             case 3:
               printf("请输入删除元素位置: ");
               scanf("%d",&k);
               j=Delete_SeqList(L,k);
               if(j!=0)
               {printf("删除后顺序表如下所示\n");Display_SeqList(L);printf("\n");}
```

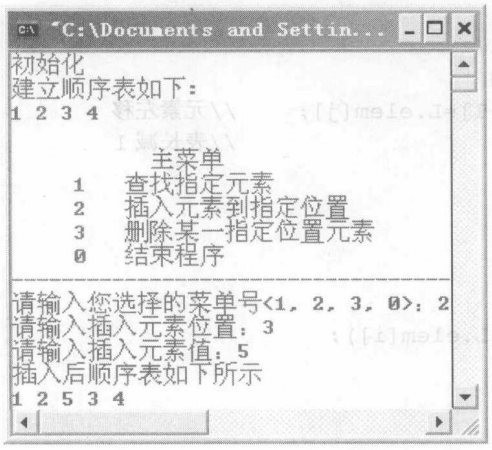


图 1-4 顺序表各种基本运算运行结果

```
            break;
        case 0:
            exit(0);break;
        default:
        printf("输入有误!");
        }
    }
}
```

程序分析：这一程序的功能是实现顺序表的各种基本运算，使用一维数组存储。程序中首先使用尾插法建立了顺序表；实现了按值查找顺序表中的某一数据元素；在顺序表的某一指定位置处插入数据元素；删除顺序表某一指定位置上的数据元素。程序运行结果如图 1-4 所示。

实验 2.2 实现单链表（线性链表）各种基本运算

考核内容：

（1）以单链表作为存储结构；

（2）实现单链表上的数据元素的插入运算；

（3）实现单链表上的数据元素的删除运算；

（4）实现单链表上的数据元素的查找。

参考程序：

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#define flag 0
typedef int ElemType;
typedef struct Lnode
{// 链式存储结构定义
    int data;                              // 定义数据域
    struct Lnode *next;                    // 定义指针域
}Lnode,*LinkList;                          // 定义结点和头指针类型名
Lnode *Get_LinkList(LinkList L, int i)
{// 在带头结点单链表中查找第 i 个数据元素,找到返回其指针,否则返回空
    Lnode *p=L;
    int j=0;
    while(p!=NULL&&j<i)  { p=p->next;j++;}
    return p;
}
int Locate_LinkList(LinkList L,int x)
{// 在带头结点单链表中查找值等于给定值的结点
    LinkList p;int j=1;
    p=L->next;
    while(p!=NULL&&p->data!=x)             //若首结点不是查找结点且可以链接到下一结点
```