高等职业教育通用教材

数据结构实验指导与测试

主编　许乐平　副主编　王琳芳

中央广播电视大学出版社

高等职业教育通用教材

数据结构实验指导与测试

主　编　许乐平
副主编　王琳芳

中央广播电视大学出版社
北　京

图书在版编目（CIP）数据

数据结构实验指导与测试／许乐平主编．—北京：中央广播电视大学出版社，2007.7

高等职业教育通用教材

ISBN 978-7-304-03889-2

Ⅰ.数… Ⅱ.许… Ⅲ.数据结构-高等学校：技术学校-教学参考资料 Ⅳ.TP311.12

中国版本图书馆 CIP 数据核字（2007）第 114491 号

高等职业教育通用教材

数据结构实验指导与测试

主　编　许乐平

副主编　王琳芳

出版·发行：中央广播电视大学出版社

电话：发行部：010-58840200

总编室：010-68182524

网址：http://www.crtvup.com.cn

地址：北京市海淀区西四环中路 45 号　**邮编**：100039

经销：新华书店北京发行所

策划编辑：徐东丽　　**责任编辑**：郭振欣

印刷：北京博图彩色印刷有限公司　　**印数**：0001~2000

版本：2007 年 8 月第 1 版　　2007 年 8 月第 1 次印刷

开本：B5　　**印张**：12　　**字数**：221 千字

书号：ISBN 978-7-304-03889-2

定价：18.00 元

全国广播电视大学职业教育
教材建设委员会

前　言

本书是与《数据结构——C ++ 描述》（中央广播电视大学出版社出版）配套使用的上机实验指导与测试。全书由三部分组成：第一部分为基础实验、第二部分为综合实验、第三部分为复习指导及模拟试题。

第一部分：按照主教材8章的内容依次给出了8个基础实验，每个实验由3～4个程序组成。考虑到课时数少同时又需加强对学生的独立设计的能力的培养，所以8个实验都给出了大部分的源代码，空缺了部分关键源代码，由学生在理解算法和数据结构的基础上读懂程序，然后将程序填写完整并且在Microsoft Visual C ++ 环境中运行和测试。各教学班可根据课时数和学生的实际情况，从每个基础实验中选择2～3个程序完成。

第二部分：由4个综合实验组成，它们可以作为数据结构课程和C ++ 语言程序设计课程的课程设计或者大作业的内容。

第三部分：由复习指导与5套模拟试题（含部分参考答案和解题提示）组成，供同学们复习和巩固所学的知识之用。

在本书的最后还给出了书写实验报告的参考规范，供同学们在撰写报告时参考。与本书配套使用的光盘给出了书中全部实验的源程序和5套模拟试题。

本教材由全国广播电视大学职业教育教材建设委员会组织编写。

本书主编单位是江苏广播电视大学。许乐平任主编，王琳芳任副主编，袁桂霞、许小媛参编。编写分工：许乐平编写第一部分的实验1和第三部分；王琳芳编写第一部分的实验2、实验3、实验4，第二部分和附录；袁桂霞编写第一部分的实验5和实验6；许小媛编写第一部分的实验7和实验8。全书由许乐平、王琳芳修订，许乐平统稿。

在本书的编写过程中得到吴汉德、吴进、刘兴武、黄雁、束正煌、臧亚琴

等领导和同仁的帮助与支持，在此一并表示衷心的感谢。

由于作者水平有限，书中难免有疏漏之处，殷切希望读者和同行专家批评指正。

联系地址：

许乐平 xulp@jstvu.edu.cn

王琳芳 wanglf686@sina.com

袁桂霞 yuangx@jstvu.edu.cn

许小媛 xuxy@jstvu.edu.cn

《数据结构》编写组

2007年5月

目　录

第一部分 基础实验

实验1　顺序表的基本操作

程序一　顺序表的建立

一、实验内容与要求

从键盘上输入一批整数，以 -999 为结束输入标志，建立一个顺序存储的线性表，然后依次输出该线性表的全部数据，最后再输出表长。

二、知识点与实现提示

顺序表的特点是：数据元素按逻辑次序依次存放到一组地址连续的存储单元里，因此表中逻辑结构上相邻的数据元素，其物理位置也一定相邻，所以可以利用这一特点，通过数组的下标来直接访问数据元素。

顺序表的结构类型定义如下：

```
const int MaxSize = 100; /* MaxSize 为线性表可能达到的最大长度 */
struct SeqList
    {
      ElemType list [MaxSize]; /* 用一维数组 list 存储线性表 */
      int size; /* 存储线性表的长度，其值≤MaxSize，空表时置为 0 */
      };
```

注意在 C ++ 语言中，数组下标是从 0 计数的，所以数据元素的序号和数组的下标相差 1。

三、参考源程序

请在下面的程序中的下划线部分填入适当的语句和文字并运行。

```
#include <iostream.h>
#include <stdlib.h>
typedef int ElemType;
const int MaxSize=100; /*MaxSize为线性表可能达到的最大长度*/
struct SeqList
{
    ElemType list[MaxSize];     /*用一维数组list存储线性表*/
    int _____;/*存储线性表的长度，其值≤MaxSize，空表时置为0*/
};

void InitList(SeqList &L)  /*初始化顺序表*/
{_____;}

int ListLength(SeqList L)  /*求线性表的长度*/
{return _____;}

void OutSeqList(SeqList L)     /*输出线性表的数据元素*/
{
    for(int i=0; i<L.size; i++)
      cout<<_____<<" ";
    cout<<endl;
}

void CreateSeqList(SeqList &L)  /*建立顺序表*/
{
    int e, i=0;
    InitList(L);            /*初始化顺序表*/
    cout<<"请按顺序输入一批整数，以-999作为结束标志："<<endl;
    cin>>e;
    while(e!=-999)    /*不是结束标志则继续执行*/
    {
        L.list[i]=____;
```

```
        i ++ ;
        if( i > MaxSize )
        { cout << "数据元素的数量超出顺序表的范围" << endl;
          exit(1); }
        cin >> e;
    }/ * while 结束 * /
    L. size = i;
}

void main( )
{
    SeqList   a;  / * 定义一顺序存储的线性表 a * /
    cout << endl << "建立的顺序表" << endl;
    cout << "--------------------------" << endl;
    CreateSeqList( a );
    OutSeqList( a );                / * 输出顺序表 * /
    cout << "表长:  " << ListLength( a ) << endl;
}
```

测试和运行结果如图 1 -1 -1 所示。

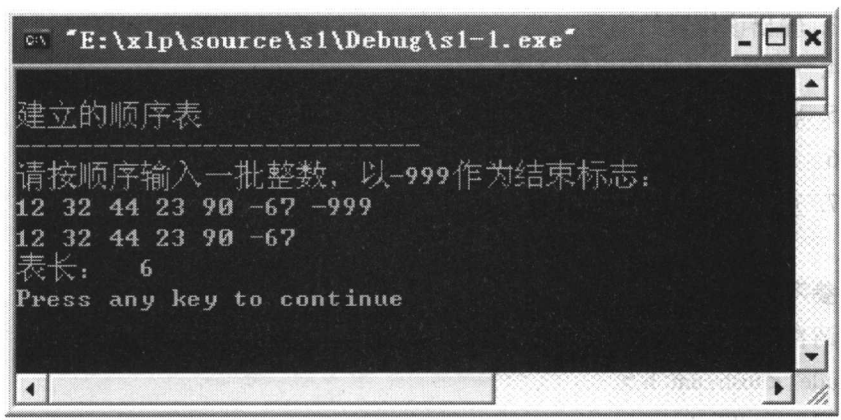

图 1 -1 -1　程序一的测试和运行结果

四、思考与提高

如果采用不断调用主教材第 32 页上的 InsList （SeqList &L, int i, Elem-

Type e）算法来建立顺序表，应当如何修改上述程序？

程序二 约瑟夫问题的求解 1

一、实验内容与要求

约瑟夫问题的一种描述是：编号为 1，2，…，n 的 n 个人顺时针地围坐成一圈，从编号为 1 的人开始顺时针从 1 开始递增报数，报到 m 时停止报数，报 m 的人出列，从他在顺时针方向上的下一个人开始重新从 1 报数到 m 停止报数，报到 m 的人出列，如此下去，直至所有的人全部出列为止。

例如 m 的初值为 5，n=9，出列的顺序为：5，1，7，4，3，6，9，2，8。

二、知识点与实现提示

用顺序表存放 n 个人的编号，为了程序的精练，可以直接用一维数组 a[n]表示顺序表。用 k 表示参加报数的人数，初始 k=n，s 为起始报数人的编号，初值 s=1，报数位置 i（即数组下标）为编号减 1，初始 i=s-1，然后进行如下操作：

1. 如果 i 等于 k，就置 i=0（实现循环报数）；
2. 计算出局人的位置 i= (i+m-1)%k；
3. 输出该位置的编号；
4. i 成为新的起始报数位置；
5. 将当前位置之后的人都前移一个位置（即表示一人出局）；
6. 参加报数的人数 k 减 1；
7. 重复上述 6 个步骤，直到 k=0 为止。

三、参考源程序

请在下面的程序中的下划线部分填入适当的语句和文字并运行。

```
#include <iostream.h>
#include <stdlib.h>
const int Maxsize=100;
typedef int ElemType;
void Josephus(int a[ ], int n, int s, int m)
{
```

```
    int  i, j, k;
    i=s-1;    /*i为报数起始位置的下标*/
    if(m<=0)
    {
      cout<<"报数不能为零或负数!"<<endl;
      return;
    }
    for(k=n; k>=1; k--)
    {        /*逐个出局，执行n次*/
      if(i==k) i=0;
      i=_____;     /*i为出局者的位置*/
      cout<<a[i]<<' ';
      for(j=i+1;j<=k-1;j++)
        _____; /*将a[i+1]到a[k-1]向前移动一个位
        置*/
    }
}

void  main()
{
    ElemType  a[Maxsize];
    int  i, n, m;
    cout<<"请输入人数:";
    _____;
    if(n>Maxsize)
    {  cout<<"人数太多!  "<<endl;
       exit(1);
    }
    for(i=0; i<n; i++)
        a[i]=i+1;  /*a[0]中放序号1, ……, a[i]中存放序号i+1*/
    cout<<"请输入报数:";
    cin>>m;
    i=1;
```

```
    Josephus(a, n, i, m); /*i是起始位置，n是座位数，m是当前报数*/
    cout << endl;
}
```

测试与运行结果如图1-1-2所示。

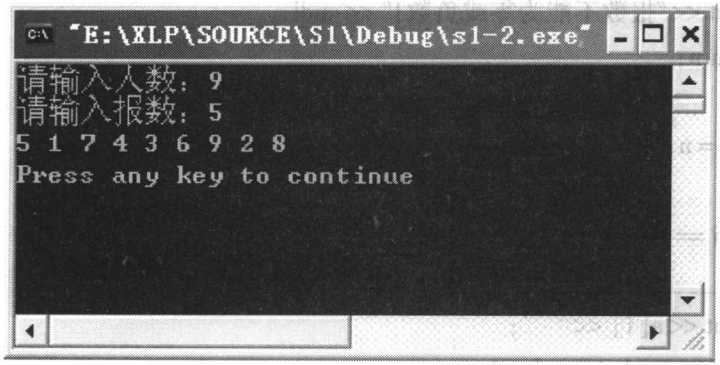

图1-1-2 程序二的测试和运行结果

四、思考与提高

解决本问题的另一种算法可以是：用一维数组a［n］存放n个人的编号，即a［i］=i+1，从开始报数位置起用计数器count计数，依次循环地读取数组元素，先判断其值是否为“-1”，如果是“-1”不计数，反之计数器加1，直到count等于m为止，当前元素就是出局者，输出其值，然后将其值赋为-1表示已出局，当前位置成为新的报数起始位置，重复上述过程直到全部元素都出局为止。

试用本算法实现求解。

程序三 约瑟夫问题的求解2

一、实验内容与要求

约瑟夫问题的另一种描述是：编号为1，2，…，n的n个人围坐一圈，每人持有一个密码（自然数）。开始任选一个自然数m作为初始报数，从编号为1的人开始顺时针从1开始递增报数，报到m时停止报数，报m的人出列，将他的密码作为新的m值，从他在顺时针方向上的下一个人开始重新从1报数，如此下去，直至所有的人全部出列为止。

例如 m 的初值为 17，n = 6，6 个人的密码依次是：2，7，9，11，4，5；出列的顺序为 5，3，4，1，6，2。

二、知识点与实现提示

程序三比程序二稍难一些，主要是每次的报数不再是定值 m，而是取决于刚出局者的密码，由它成为新的报数。因此可在一维数组 a［n］存放 n 个人的密码，m 为报数，初始报数从键盘输入。设置计数器 count 用于统计有效报数的人数，每次重新开始报数时清为 0。i 为起始报数前一个位置的下标，初值为 -1，然后执行如下操作：

1. 循环后移一个位置，即 i = (i + 1)% n;
2. 只要当前位置上的数据 a［i］不为“-1”，count ++;
3. 重复上述 1 和 2 两步直到 count 等于 m，当前 a［i］为出局者，将其编号（即 i + 1）输出；
4. 将 a［i］的值赋给 m 成为下一轮的报数；
5. 将 a［i］置为“-1”，表示其已出局；
6. 计数器 count 置 1；
7. 重复上述 1 ~ 6 步，直到全部都出局为止。

三、参考源程序

请在下面的程序中的下划线部分填入适当的语句和文字并运行。

```
#include <iostream.h>
#include <stdlib.h>
const int Maxsize = 100;
typedef int ElemType;
void Josephus(int a[ ], int s, int n, int m)
{/*s 为起始座位号(1 ~ n)，n 为人数，m 起始报数*/
    int i = s - 2; /*i 为下标，由于下面 while 循环中先做 i + 1 操作，所以
                     这里需先扣除 1*/
    for(int j = 1; j <= _____; j ++)/*实现 n 个人出局*/
    {
        int count = 1;
        while(count <= m)/*只要不是 -1 进行计数，直到第 m 个人*/
        {
```

```
            i = (i + 1) % n;
            if(a[i] ! = _____) count ++ ;
        }/ * while 结束 * /
        cout << i + 1 << ' '; / * 输出当前出局者的序号 * /
        m = _____; / * 当前出局者的密码成为新的报数 * /
        a[i] = -1; / * 当前出局者置为 -1，表示已出局，当前位置作为下
                    一轮起始前一个位置 * /
    }/ * for 结束 * /
}
void  main( )
{
    ElemType  a[Maxsize];
    int  i, n, m;
    cout << "请输入人数: ";
    _____;
    if(n > Maxsize)
    {  cout << "人数太多!" << endl;
       exit(1);
     };
    for(i = 1; i < = n; i ++ )/ * 为圈子里的人安排密码 * /
    {
       cout << "请输入第" << i << "人的密码: ";
       cin >> _____;
    }/ * for 结束 * /
    cout << "请输入初始报数: ";
    cin >> m;
    Josephus(a, 1, n, m);
    cout << endl;
}
```

测试与运行结果如图 1－1－3 所示。

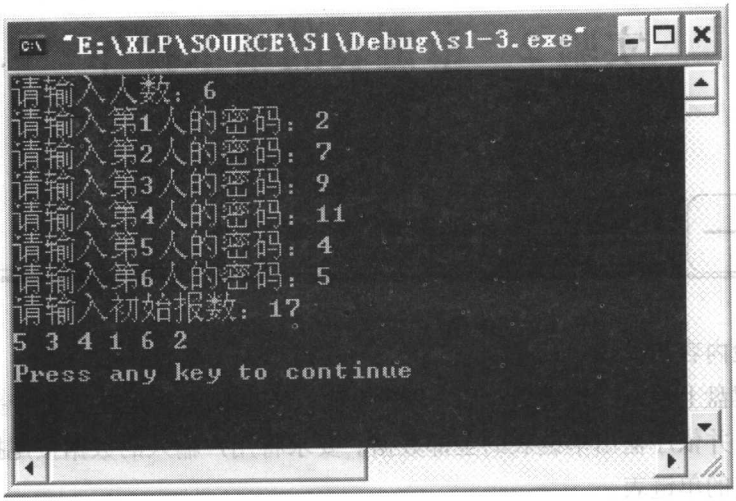

图 1－1－3　程序三的测试和运行结果

四、思考与提高

对上述约瑟夫问题，试采用循环单链表作为存储结构来实现求解。