中国地质大学(武汉)实验教学系列教材
中国地质大学(武汉)实验技术研究项目资助

数据结构实习指导书

SHUJU JIEGOU SHIXI ZHIDAOSHU

李桂玲 ◎主　编

朱晓莲
郭　艳 ◎副主编

中国地质大学出版社
ZHONGGUO DIZHI DAXUE CHUBANSHE

数据结构实习指导书

SHUJU JIEGOU SHIXI ZHIDAOSHU

主　编：李桂玲

副主编：朱晓莲　郭　艳

中国地质大学出版社

ZHONGGUO DIZHI DAXUE CHUBANSHE

图书在版编目(CIP)数据

数据结构实习指导书/李桂玲主编;朱晓莲,郭艳副主编. —武汉:中国地质大学出版社,2014.11

中国地质大学(武汉)实验教学系列教材

ISBN 978-7-5625-3552-2

Ⅰ.①数…

Ⅱ.①李…②朱…③郭…

Ⅲ.①数据结构-高等学校-教学参考资料

Ⅳ.①TP311.12

中国版本图书馆 CIP 数据核字(2014)第 252772 号

数据结构实习指导书

李桂玲 **主 编**

朱晓莲 郭 艳 **副主编**

责任编辑:马 严　　责任校对:张咏梅

出版发行:中国地质大学出版社(武汉市洪山区鲁磨路 388 号)　　邮政编码:430074

电 话:(027)67883511　　传真:67883580　　E-mail:cbb@cug.edu.cn

经 销:全国新华书店　　http://www.cugp.cug.edu.cn

开本:787 毫米×1 092 毫米 1/16　　字数:288 千字　印张:11.25

版次:2014 年 11 月第 1 版　　印次:2014 年 11 月第 1 次印刷

印刷:荆州鸿盛印务有限公司　　印数:1—1 000 册

ISBN 978-7-5625-3552-2　　定价:22.00 元

前　言

数据结构课程是计算机学科一门非常重要的专业基础课，广泛应用于信息科学、系统工程、应用数学及各种工程技术领域，其实践性较强，对学生的动手能力要求较高。数据结构课程实践环节的教学效果影响着学生后继专业课程的学习、就业、科研等方面。

美国 ACM/IEEE 组织一直对世界计算机教育起着指导性作用。2010 年 12 月，ACM/IEEE CS 2013 工作组对全球计算机系的系主任进行问卷调查，反馈表明，关于计算机专业核心知识，排名前三的是计算思维、程序设计基础、数据结构。计算机专业委员会的调研也显示，IT 项目主管认为排在头三位的专业技能分别是编程能力、数据结构和算法。

鉴于数据结构在计算机专业中的核心地位，以及其在就业中属于必备专业技能的重要性，数据结构课程的实践教学环节的建设具有重要的研究价值和实际意义。目前我校不仅计算机学院所有专业开设了数据结构课程，其他学院与信息技术相关的专业也开设了数据结构课程。数据结构是一门实践性很强的课程，绝大多数高校都开设了数据结构实验课程。

数据结构课程的上机实习包括两个部分：课程内的上机实习和课程结束后的数据结构课程设计。通过本课程的上机实习，要求在数据结构的逻辑特性和物理表示、数据结构的选择和应用、算法的设计及其实现等方面加深对课程基本内容的理解。同时，在程序设计方法以及上机操作等基本技能和科学作风方面受到比较系统和严格的训练。

本实习书的宗旨是“提高学生动手能力，培养学生创新能力”，主线为“课程实习范例—课程实习内容—课程设计范例—课程设计内容”。课程内的上机实习是课程设计阶段的基础，课程设计是课程内的上机实习的巩固和深化、拓展和延伸。在课程内的上机实习部分，每章先总结该章的基本知识，接着给出该章的上机实习示例，然后给出该章的上机实习题目；在课程设计部分，先给出课程设计的基本

要求，接着详细给出数据结构课程设计的示例，然后给出数据结构课程设计题目。

本书遵循数据结构教学大纲的要求，内容共9章，可分为4个部分。第一部分为绪论即第一章，介绍本实习书的要求和目标，后面8章的内容给出了实习报告示例。第二部分为数据结构课程内的上机实习，按照课程内容的教学顺序“线性结构—非线性结构—查找和排序”进行组织，具体为第二章到第七章。其中，第二、第三章介绍线性结构，分别介绍线性表、堆栈和队列；第四、第五章介绍非线性结构，分别介绍树和二叉树、图；第六、第七章介绍两类常用的操作，分别为查找、内部排序。第三部分为数据结构课程设计，即第八章。第四部分为数据结构试题及答案，即第九章，给出了两套历年的数据结构考试试题及答案。书中第一至第八章每章都给出了具体的实习示例，并且示例的代码在Visual C++ 6.0环境下全部调试运行通过。在教师教学和学生学习的过程中，关于上机实习题目的选择，可由教师结合教学学时和教学要求在宏观上进行把关，实际的题目可由学生根据实习目的和要求选择具体题目。

本书内容编写分工为：朱晓莲编写第六、第七章；郭艳编写第四、第五章，其中2013级研究生王鑫协助这两章示例代码的实现与测试；李桂玲、朱晓莲共同编写第一章；第八章第三节数据结构课程设计的题目由三位编者共同收集整理；其余章节均由李桂玲编写。本书编写的过程中吸取了许多前辈的成果，也借鉴和引用了一些文献。在此，对为本书直接或间接付出辛苦劳动的同仁表示深深的感谢！

由于编者水平有限，书中难免存在不足之处，希望各位读者能够批评指正，以提高本书的水平。

编　者

2014年8月

目　录

第一章 绪 论

第一节 本书目标及内容

一、本书目标

数据结构课程是计算机学科一门非常重要的专业基础课。广泛应用于信息科学、系统工程、应用数学及各种工程技术领域，其实践性较强，对学生动手能力要求较高。由于该课程是一门实践性较强的软件基础课程，为了学好这门课程，每个学生必须完成一定数量的上机实习作业。开设实验课的目的就是为了达到理论与实际应用相结合。

数据结构的上机实习包括两个部分：课程内的上机实习和数据结构课程设计。通过本课程的实习，要求在数据结构的逻辑特性和物理表示、数据结构的选择和应用、算法的设计及其实现等方面加深对课程基本内容的理解。同时，在程序设计方法以及上机操作等基本技能和科学作风方面受到比较系统和严格的训练。

二、本文内容安排

本书共分为9章，内容组织如下：

第一章为绪论。介绍本书和实习的目的和要求，后面8章的内容导引，给出了实习规范和要求，并以约瑟夫环问题为例给出了实习报告示例。

第二章为线性表。先介绍了线性表的基本知识，包括线性表的定义和特征、线性表的顺序存储结构及操作的实现、线性表的链式存储结构及操作的实现；接着给出了上机实习示例一元稀疏多项式的加法运算，该题目是线性表的链式存储结构的经典应用；后面列出了线性表的上机题目。

第三章为堆栈和队列。先介绍了堆栈和队列的基本知识，主要包括堆栈的定义及特征、堆栈的两种存储结构及操作的实现、队列的定义及特征、队列的两种存储结构及操作的实现；接着给出了上机实习示例电话客户服务模拟，该题目涉及队列的应用；后面列出了堆栈和队列的上机题目。

第四章为树和二叉树。先介绍了树和二叉树的基本知识，包括树的定义和存储结构，二叉树的定义、性质、存储结构及遍历操作的实现等；接着给出了上机实习示例高校社团管理，该题目是高校中常见的学生组织关于树的应用；后面列出了树和二叉树的上机题目。

第五章为图。先介绍了图的基本知识，包括图的定义、存储结构、图的遍历算法和图的应

用;接着给出了上机实习示例校园局域网布线问题,校园局域网的布线问题可看成图问题来解决;后面列出了图的上机题目。

第六章为查找。先介绍了查找的基本知识,包括静态查找表、动态查找表和哈希表;接着给出了上机实习示例二叉排序树的实现,该题目演示了二叉排序树的主要操作;后面列出了查找的上机题目。

第七章为内部排序。先介绍了内部排序的基本知识,包括插入排序、选择排序、交换排序、归并排序和基数排序;接着给出了上机实习示例直接插入排序基于单链表的实现;后面列出了内部排序的上机题目。

第八章为数据结构课程设计。先介绍了数据结构课程设计要求;接着重点给出了一个课程设计示例计算命题演算公式的真值,该题目涉及到线性结构(堆栈)和非线性结构(二叉树)的综合应用;然后列出了一些数据结构课程设计的题目。

第九章给出了两套历年数据结构期末考试的试题及答案。

第二节 实习规范及要求

一、实习步骤

(1)按照每次实习的要求选定题目。

(2)分析题目要求,明确要实现的功能;设计数据结构;根据结构化程序设计方法,先确定算法的基本思想、主要模块、各模块功能及调用关系,再用C语言对主要算法进行细化。

(3)编写好上机源程序,并进行静态检查,使程序中的逻辑错误和语法错误减少到最低程度。

(4)了解所用机器的操作规程,考虑好调试手段和测试数据等,做好上机准备。

(5)上机调试程序,程序通过后,打印一份源程序清单及对测试数据的运行结果。

(6)最后写好调试分析报告及使用说明,完成一份完整的实习报告。在调试分析中应总结上机中遇到的问题及解决方法;对所设计的算法的时间复杂度和空间复杂度进行分析;算法的进一步改进;本次实习的心得体会等。使用说明主要描述如何使用你的程序以及使用时的注意事项。

二、实习报告的内容

1. 需求分析

描述问题,简述题目要解决的问题是什么?规定软件做什么?原题条件不足时补全。

2. 设计

(1)设计思想:存储结构(题目中限定的要复述);主要算法基本思想。

(2)设计表示:每个函数或过程的头和规格说明;列出每个过程或函数所调用和被调用的过程或函数,也可以通过调用关系图表达。

(3)实现注释:各项功能的实现程度、在完成基本要求的基础上还实现了什么功能。

(4)详细设计表示：主要算法的实现。

3. 调试分析

调试过程中遇到的主要问题是如何解决的；对设计和编码的回顾讨论和分析；时间和空间复杂度的分析；改进设想；经验和体会等。

4. 用户手册

即使用说明。

5. 测试数据及结果

如果题目规定了测试数据，则结果要包含这些测试数据和运行输出，当然还可以包括其他测试数据及运行输出结果(有时需要多组数据)。

6. 源程序清单

源程序关键之处均需要添加必要的注释。

第三节 实习报告示例：约瑟夫环问题

【问题描述】

约瑟夫环问题的一种描述是：编号为1，2，…，n 的 n 个人按顺时针方向围坐一圈，每人持有一个密码(正整数)。一开始任选一个正整数作为报数上限值 m，从第一个人开始按顺时针方向自1开始报数，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，从他在顺时针方向的下一人开始重新从1报数，如此下去，直至所有的人全部出列为止。试设计一个程序求出出列顺序。

【基本要求】

利用单向循环链表存储结构模拟此过程，按照出列的顺序输出各人的编号。

【测试数据】

m 的初值为20，n=7，7个人的密码依次是3，1，7，2，4，8，4(正确的出列顺序应为6，1，4，7，2，3，5)。

【实现提示】

程序运行后，首先要求用户指定初始报数上限值，然后读取各人的密码，可设 n≤30。此题所用的循环链表不需要“头结点”，请注意空表和非空表的界限。

实习报告

题目：约瑟夫环问题

一、需求分析

(1)要求以循环链表模拟约瑟夫环，结点个数(人数)n≤30。各人的密码和初始报数上限值必须为正整数。

(2)程序以人机对话的方式执行，即在计算机终端上显示“提示信息”后，由用户在键盘上

输入相应的运算命令或数据；相应的输入数据和运算结果显示在其后。

(3)测试数据：n＝7，m＝20，7 个结点的密码分别为 3，1，7，2，4，8，4，输出结果应为 6，1，4，7，2，3，5。用 m＝1 再测试一次，输出结果应为 1，4，6，5，7，3，2。

二、设计

1. 设计思想

1)存储结构

根据问题描述，可采用循环单链表结构。循环单链表的结点结构如图 1－1 所示。

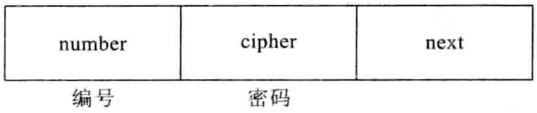

图 1－1　结点结构

结点定义如下：

```
typedef struct nodetype
{int number;
 int cipher;
 struct nodetype *next;
}clinktp;
```

为便于操作，可建立由尾指针 ra 指示的循环链表，如图 1－2 所示。

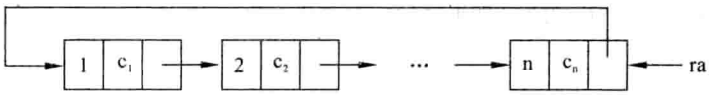

图 1－2　循环单链表示意图

2)基本思想

(1)建立一个由尾指针 ra 指示的有 n 个结点的约瑟夫单循环链表。

(2)根据给定的 m 值，寻找第 m 个结点(应该找到第 m－1 个结点的地址才便于删除)，输出该结点的 number 值，把该结点的 cipher 值赋给 m，作为下一次循环的报数上限 m，删除第 m 个结点。如此循环 n 次(实际只需循环 n－1 次，最后只剩一个结点时，直接输出即可)结束。

2. 设计表示法

1)过程或函数调用关系图

main( )→createlist( )→locfor( )

2)基于数据结构的操作组

本次实习的题目比较简单，只用到了一个循环链表，函数 createlist( )，locfor( )都基于循环链表 ra。

3)过程或函数接口规格说明

```
int createlist(clinktp **ra,int n);   /*建立以 ra 为尾指针的有 n 个结点的循环链表*/
```

```
void locfor(clinktp *ra,int m);  /*在循环链表中根据要求输出各次从链表中删除的结点序号*/
```

3. 实现注释

(1)根据输入的 n 值建立循环链表。

(2)可任意输入初始报数上限值 m(正整数)。

4. 详细设计

```
/*在循环链表中根据报数上限值输出从链表中删除的结点序号*/
void locfor(clinktp*ra,int m)
{
    int i;
    clinktp *p,*pre,*s;/*p 指针用于指向第 m 个结点,pre 指向 p 的直接前驱结点*/
    pre=ra;
    p=ra->next;
    while (pre!=p)
    {
        for(i=1;i<m;i++ )  /*循环找到第 m 个结点*/
        {  pre=p;p=p->next;}
        printf("%d\n",p->number);/*输出将出队的人的编号*/
        m=p->cipher;  /*将出队的人的密码作为下一次报数的上限值*/
        pre->next=p->next;  /*接着此人出队,执行删除操作*/
        s=p;
        p=p->next;
        free(s);
    }
    printf("%d\n",p->number);  /*输出最后一个出队的人的编号*/
    free(p);
}
```

三、调试分析

(1)刚开始时忽略了变量参数的传递问题,使调试程序时费时不少,今后应重视确定参数的变量和赋值属性的区分和标识。

(2)由于考虑不周,开始建立的是头指针指示的循环链表,导致当 m=1 时,得不到正确结果。后来增加一条 for 循环语句,搜索到表尾结点,可得到正确结果,但影响算法的效率。改进的方法是建立由尾指针指示的循环链表。

(3)createlist( ):时间复杂度:O(n),空间复杂度:O(n)

locfor( ):时间复杂度:O(n×m),空间复杂度:O(n)

(4)本次实习作业采用结构化的程序设计方法,将程序划分为主函数、建循环链表、求出出列顺序等几个模块,使得设计时思路清晰,实现时调试顺利,各模块具有较好的可重用性,确实得到了一次良好的程序设计训练。

四、用户手册

(1)本程序的运行环境为 Visual C++6.0,可执行文件为 joseph.exe。
(2)进入演示程序后,即出现提示信息:
input n:_
输入结点个数 n 值后,出现提示信息:
input cipher:_
然后依次输入 n 个结点的密码,此时出现提示信息:
input m:_
输入报数上限值 m 后,程序执行相应操作后,显示相应结果。

五、测试数据及结果

---约瑟夫环问题---
input n:7

i nput cipher:
3 1 7 2 4 8 4

i nput m:20

出队的编号顺序为:
6
1
4
7
2
3
5

六、源程序清单

```
/*文件名:joseph.c*/
# include <stdio.h>
# include <stdlib.h>

typedef struct nodetype
{
    int number;      //编号
    int cipher;      //密码
    struct nodetype *next;   //后继指针域
```

```
}clinktp;/*定义循环链表的结点结构*/

/*建立以 ra 为尾指针的有 n 个结点的循环链表,链表建好后不带头结点*/
int createlist(clinktp **ra,int n)
{
    int i;
    clinktp *p,*q,*la;
    if ((la=(clinktp*)malloc(sizeof(clinktp)))==NULL)
    {
        printf("\noverflow!");
        return 0;
    }
    printf("\ninput cipher:\n");
    q=la;
    for(i=1;i<=n;i++)      /*循环 n 次创建循环单向链表*/
    {
        if ((p=( clinktp *)malloc(sizeof(clinktp )))==NULL)
        {
            printf("\noverflow!");
            return 0;
        }
        p->number=i;
        scanf("%d",&p->cipher);
        q->next=p;
        q=p;
    }
    q->next=la->next;
    free(la);
    (*ra)=q;
    return 1;
}

/*在循环链表中根据报数上限值输出从链表中删除的结点序号*/
void locfor(clinktp *ra,int m)
{
    int i;
    clinktp*p,*pre,*s;/*p 指针用于指向第 m 个结点,pre 指向 p 的直接前驱结点*/
    pre=ra;
    p=ra->next;
```

```
    while(pre!=p)
    {
        for(i=1;i<m;i++)  /*循环找到第 m 个结点*/
        {pre=p;p=p->next;}
        printf("%d\n",p->number);/*输出将出队的人的编号*/
        m=p->cipher;      /*将出队的人的密码作为下一次报数的上限值*/
        pre->next=p->next;  /*接着此人出队,执行删除操作*/
        s=p;
        p=p->next;
        free(s);
    }
    printf("%d\n",p->number);  /*输出最后一个出队的人的编号*/
    free(p);
}

void main( )
{
    int m,n;
    clinktp *ra;
    printf("---约瑟夫环问题---\n");
    printf("input n:");
    scanf("%d",&n);
    if (!createlist(&ra,n))
        exit(1);
    printf("\ninput m:");
    scanf("%d",&m);
    printf("\n 出队的编号顺序为:\n");
    locfor(ra,m);
}
```

第二章　线性表

线性结构是最简单常用的数据结构。线性表是一种典型的线性结构。线性结构的主要特征有：存在唯一的第一个元素，该元素无直接前驱；存在唯一的最后一个元素，该元素无直接后继；除第一个元素和最后一个元素外，每个元素都有唯一的直接前驱和唯一的直接后继。

在现实生活中，有许多逻辑结构是线性表的应用。例如，图书馆的图书自动检索系统中的图书信息表、单位的工资管理系统中的工资信息表等都可以在系统中用线性表进行表示。

本章的主要知识点有：线性表的定义及特征、线性表的顺序存储结构及操作的实现、线性表的链式存储结构及操作的实现。

第一节　基本知识

一、线性表的定义

线性表是由 $n(n\geqslant 0)$ 个元素 $a_0, a_1, a_2, \cdots, a_{n-1}$ 组成的有限序列。其中，n 为数据元素的个数，也称为表的长度。当 $n=0$ 时称为空表；当 $n>0$ 时为非空的线性表，记为 $(a_0, a_1, a_2, \cdots, a_{n-1})$。

线性表的特性主要有：线性表中所有数据元素的类型都是一致的；数据元素在线性表中的位置只取决于它的序号；数据元素之间的逻辑关系是线性的。

线性表的抽象数据类型定义为：

```
ADT Linear_list
    数据对象：D={ai | ai∈ElemSet，i=0，1，2，…，n－1，n≥0}
    数据关系：S={<ai-1，ai> | ai-1，ai∈D，i=1，…，n－1}
    基本操作：设 L 为 Linear_list 型
            InitList(L)          //初始化空表操作；
            ListLength(L)        //求表长函数；
            ListGet(L，i)        //取序号为 i 的数据元素；
            LocateElem(L，e)     //定位函数；
            ListInsert(L，i，e)  //插入操作；
            ListDelete(L，i)     //删除操作；
```

二、线性表的顺序存储结构

用一组地址连续的存储单元依次存放线性表的数据元素称为线性表的顺序存储结构，简称为顺序表。顺序表的特点是，逻辑上相邻的数据元素在物理存储位置上也相邻。

1. 顺序表的描述

```
#define Maxsize <顺序表的最大元素个数>
typedef struct
{DataType list[Maxsize];
    int size;
}SeqList;
```

其中，DataType 根据实际要解决的问题确定具体的数据类型，list 数组用于存储线性表的数据元素，size 表示顺序表中当前存储的数据元素的个数。

2. 顺序表的初始化操作的实现

```
void InitList(SeqList *L)
{L ->size=0;}
```

3. 顺序表的插入操作的实现

```
int ListInsert(SeqList *L,int i,DataType x)
  /*在顺序表 L 中的第 i 个元素之前插入一个新的数据元素 x*/
{  int  j;
    if  (L ->size>=MaxSize)
        {  printf("表已满,不能插入!\n");return 0;}
    else if (i<0||i>L ->size)
        {  printf("i 值不合法!\n");          return 0;}
    else
        {  for (j=L ->size;j>i;j--)
             L ->list[j]=L ->list[j -1];   /*数据元素依次后移*/
        L ->list[i]=x;   /*插入 x*/
        L ->size++;
        return 1;}
}
```

4. 顺序表的删除操作的实现

```
int ListDelete(SeqList *L,int i,DataType *x)
  /*删除顺序表 L 的第 i 个数据元素*/
{  int j;
    if (L ->size<=0)
        {  printf("表空,不能删除!\n");   return 0;   }
    else   if (i<0||i>L ->size -1)
        {  printf("i 值不合法!\n");          return 0;   }
    else{     *x=L ->list[i];
              for (j=i+1;j<=L ->size -1;j++)
                    L ->list[j -1]=L ->list[j];   /*数据元素依次前移*/
        L ->size--;
```