主 编 梁金明 鲜乾坤

数据结构

实验指导与习题

SHUJU JIEGOU

SHIYAN ZHIDAO YU XITI

电子科技大学出版社

数据结构
实验指导与习题

主编　梁金明　鲜乾坤

电子科技大学出版社

图书在版编目（CIP）数据

数据结构实验指导与习题 / 梁金明, 鲜乾坤主编
. -- 成都 : 电子科技大学出版社, 2014.8

ISBN 978-7-5647-2580-8

Ⅰ. ①数… Ⅱ. ①梁… ②鲜… 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 195423 号

数据结构实验指导与习题

主编　梁金明　鲜乾坤

出　　版：电子科技大学出版社（成都市一环路东一段 159 号电子信息产业大厦　邮编：610051）
策划编辑：张　鹏　吴艳玲
责任编辑：张　鹏
主　　页：www.uestcp.com.cn
电子邮箱：uestcp@uestcp.com.cn
发　　行：新华书店经销
印　　刷：郫县犀浦印刷厂
成品尺寸：185mm×260mm　　印张 13.5　字数 332 千字
版　　次：2014 年 8 月第一版
印　　次：2014 年 8 月第一次印刷
书　　号：ISBN 978-7-5647-2580-8
定　　价：30.00 元

前　言

《数据结构》是计算机及相关专业的一门专业性、实践性很强的专业基础课程，为了学好本门课程，学生必须完成一定数量的上机实验与习题。通过实验与习题，进一步理解和掌握数据结构的基本知识、基本理论与基本操作技能，以及各种不同数据结构的实现算法，提高在数据结构的选择和应用、算法的设计与实现等方面的能力。

本书共分三部分：第一部分是数据结构实验指导，设计了 10 个实验单元共 28 个实验题目，涉及《数据结构》课程的线性表、栈与队列、串、数组与广义表、树与二叉树、图、查找、内部排序等章节的知识。每个实验题目包括问题描述、基本要求、数据结构设计、算法分析与设计、参考程序、测试结果、课外实验几个组成部分不等。为加强学生能力的训练，给出的参考程序并不完整，有的实验题目只给出了数据结构设计和基本算法，或只提基本要求，需要读者去分析、去设计、去完善，目的是提高实验的综合性与设计性。在实验部分的最后，给出了实验报告规范、实验报告范例两个附录。实验中的全部程序均给出了详细的注释，可以帮助读者理解和掌握实验中的各种算法，从而提高不同数据结构及算法的分析与设计能力。第二部分是数据结构习题，内容包括绪论、线性表、栈与队列、串数组和广义表、树与二叉树、图、查找、排序共 8 章内容，设计了单项选择题、填空题、判断题、应用题、算法填空题、算法设计题共 6 种题型，选编了逾 430 道题，题目难度适中。第三部分是第二部分习题的参考答案，部分习题进行了解析。

本书注重基础性、应用性、操作性、趣味性，在实验题目的设计、习题类型及难度的设置等适合二本院校的学生使用。

书中每个实验的测试结果均是在 VC++6.0 环境下的测试结果。由于书中所有的参考程序均不完整，也未提供实验答案，有需要实验答案的教师和工程技术人员可以向编者索取，但不建议直接提供给有教师指导的实验者。

尽管编者态度严谨、努力工作，但由于水平与能力的局限，书中难免有错误和不妥之处，敬请读者批评指正。

编　者

2014 年 5 月于自贡

目　　录

第一部分 数据结构实验指导 1

概　述 2

实验一　顺序表的操作及应用 4

实验题 1.1　顺序表的基本操作 4

实验题 1.2　顺序表的基本运算 10

实验题 1.3　有序顺序表的归并 14

实验二　链表的操作及应用 16

实验题 2.1　单链表的基本操作 16

实验题 2.2　线性链表的归并 23

实验题 2.3　Josephus 问题求解 26

实验题 2.4　链表逆置问题 30

实验三　栈的操作及应用 31

实验题 3.1　顺序栈的基本操作与进制转换 31

实验题 3.2　链栈的基本操作与括号匹配问题 34

实验题 3.3　求解迷宫路径问题 36

实验四　队列的操作及应用 41

实验题 4.1　链队列的基本操作 41

实验题 4.2　循环队列的基本操作 44

实验题 4.3　利用队列实现杨辉三角的输出 48

实验五　串的操作及应用 49

实验题 5.1　串的基本操作 49

实验题 5.2　串的模式匹配 55

实验六　矩阵的基本运算 61

实验题 6.1　螺旋方阵输出 61

实验题 6.2　稀疏矩阵的基本运算 63

实验七　二叉树的遍历及应用 70

实验题 7.1　二叉树的遍历 70

实验题 7.2　二叉树的基本运算 77

实验题 7.3　Huffman 树与 Huffman 编码 82
实验八　图的遍历及应用 87
实验题 8.1　图的基本操作 87
实验题 8.2　求解图的最小生成树 94
实验题 8.3　求解 AOE 网的关键路径 98
实验九　查找的操作及应用 104
实验题 9.1　静态查找表的基本操作 104
实验题 9.2　动态查找表的基本操作 109
实验题 9.3　散列运算的实现 115
实验十　内部排序的运算及应用 122
实验题 10.1　内部排序的基本运算 122
实验题 10.2　排序应用 131
附 1　《数据结构》实验报告规范 132
附 2　实验报告范例 133
第二部分　数据结构习题 142
第 1 章　绪　论 143
第 2 章　线 性 表 146
第 3 章　栈和队列 154
第 4 章　串、数组和广义表 158
第 5 章　树和二叉树 161
第 6 章　图 167
第 7 章　查　找 173
第 8 章　排　序 180
第三部分　习题参考答案 184
第 1 章　绪　论 185
第 2 章　线 性 表 185
第 3 章　栈和队列 191
第 4 章　串、数组和广义表 193
第 5 章　树和二叉树 195
第 6 章　图 200
第 7 章　查　找 204
第 8 章　排　序 208

第一部分
数据结构实验指导

概　述

一、课程目的

《数据结构》是计算机及相关专业的一门专业性、实践性很强的专业基础课程，为了学好本门课程，每个学生必须完成一定数量的上机实验。通过本课程的实验，进一步理解和掌握数据结构的基本知识、基本理论与基本操作技能，以及各种不同数据结构的实现算法，提高在数据结构的选择和应用、算法的设计与实现等方面的能力。同时，实验题中的问题比平时的练习题要复杂，也更接近实际，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。

本课程实验的目的是旨在使学生进一步巩固课堂上所学的理论知识，深化理解和灵活掌握教学内容，培养学生算法设计的能力和解决实际问题的程序设计能力。

二、实验名称与学时分配

实验单元	实验名称	建议学时
1	顺序表的操作及应用	2
2	链表的操作及应用	2
3	栈的操作及应用	2
4	队列的操作及应用	2
5	串的操作及模式匹配	2
6	矩阵的基本运算	2
7	二叉树的遍历及应用	4
8	图的遍历及应用	3-4
9	查找的操作及应用	2
10	内部排序的运算及应用	2-3

三、实验要求

1．问题分析

充分地分析和理解问题本身，弄清要求做什么，包括功能要求、性能要求、设计要求和约束条件，以及基本数据特性，数据间的联系等。

2．数据结构设计

针对要解决的问题，应考虑多种可能的数据结构，力求从中得出最佳方案(必须连同算法一起考虑)，确定主要的数据结构及全程变量。对引入的每种数据结构和全程变量要详细说明其功能、初值和操作特点。

3．算法设计

算法设计分概要设计和详细设计，概要设计着重解决程序的模块设计问题，包括如何把一个项目或一个实验题目自顶向下分解成若干功能独立的顺序模块，并决定模块的接口，即模块间的相互关系以及模块之间的信息交换问题。详细设计则要求设计每个模块内部的具体算法，包括输入、处理和输出，采用类 C 语言描述。

4．测试用例设计

准备典型测试数据和测试方案，测试数据要有代表性和敏感性，测试方案包括模块测

试和模块集成测试。

5．上机调试

对程序进行编译，纠正程序中可能出现的语法错误，测试前，先运行一遍程序看看究竟将会发生什么，如果错误较多，则根据事先设计的测试方案并结合现场情况进行错误跟踪，包括打印执行路径或输出中间变量值等手段。

注意：每一次实验，学生必须在课前完成前 4 项准备工作，并准备好源程序电子版，上机时才有充分时间进行测试和调试，以解决存在的问题。

四、实验考核

每次实验主要考核两个方面，一是现场操作技能考核，重点考核现场实验完成的数量和质量、实验结果的准确性以及实验态度等方面；二是实验报告的考核，每次实验结束后，实验者均应提交实验报告，实验报告应包括如下内容：

1．问题描述：简述题目要解决的问题是什么。

2．算法设计：包括存储结构设计、主要算法设计等。用类 C 语言或用框图描述。

3．调试报告：调试过程中遇到的问题是如何解决的，对设计和编码的讨论和分析。

4．算法分析与改进：算法的时间复杂度和空间复杂度分析，算法改进的设想。

5．经验和体会。

附：源程序清单和运行结果。

源程序要加注释。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，当然还可以含有其他测试数据和运行输出（有时需要多组数据）。

五、测评标准

1．要求从每个实验中选择实验题目，完成实验，撰写实验报告。成绩测评标准如下：

2．程序运行正确，要求的功能均能实现，实验报告全面，测试数据设计得当，测试结果分析恰当，算法效率分析正确，可获得 90 分以上。

3．要求的部分功能不能完全实现，但程序完整，实验报告全面，测试数据设计得当，测试结果分析恰当，算法效率分析正确，可获得 80～89 分。

4．程序存在一些局部问题，程序完整，实验报告基本完整，有测试有分析，可获得 70～79 分。

5．程序结构不太清晰，存在一些错误，但学习态度比较端正，实验报告基本完整，有分析，可获得 60～69 分。

6．若程序错误很多，要求的功能基本不能实现；或抄袭他人设计，则本次实验不及格。

六、实验学时

建议实验 24 学时。

实验一　顺序表的操作及应用

一、实验目的

1．掌握使用 C++/VC 环境上机调试程序的基本方法；
2．掌握线性表顺序存储的概念，加深对顺序存储数据结构的理解；
3．掌握顺序表的基本运算及简单应用问题，逐步培养解决实际问题的编程能力。

二、 注意事项：

1．在磁盘上创建一个目录，专门用于存储数据结构实验的程序。

2．建议实验者不要过分依赖本书提供的算法或参考程序，实验者对每一个实验题目可独立进行算法分析与设计，独立设计完整的上机程序。只有经过严格的训练，才能显著提高算法分析与设计能力和程序设计能力，更有助于《数据结构》课程知识体系的理解和掌握。

三、实验内容

实验题 1.1　顺序表的基本操作

1．问题描述

设有线性表（34，12，45，64，28，36，45），采用顺序存储结构。编程实现有关顺序表的下列基本操作：

（1）初始化一个空的顺序表；
（2）在顺序表的第 i 个位置上插入一个新元素；
（3）在顺序表中查找指定值的元素位置；
（4）删除顺序表中的第 i 个元素；
（5）删除顺序表中指定值的元素；
（6）输出顺序表中的所有元素值。

2．基本要求

（1）采用动态分配方式设计顺序表的存储结构；
（2）用反复执行插入操作的方式建立线性表；
（3）每完成一个步骤，必须及时输出顺序表中的所有元素，便于观察操作结果；
（4）完善参考程序，并在参考程序中的下划线处填上适当的语句或文字；
（5）设计测试用例，上机调试、测试完善后的参考程序，保存和打印测试结果，对测试结果进行分析，包括算法效率分析。

3．数据结构设计

```
#define MAXSIZE 100              // 空间初始分配量
```

```
#define INCREMENT 10          // 空间分配增量
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
} SqList;
```

4．算法分析与设计

（1）创建空顺序表

① 申请一块连续的存储空间，令 L.elem 指向该空间基地址；

② 置顺序表长度为 0；

③ 置分配的空间长度为 MAXSIZE。

（2）插入算法

将元素 x 插入在线性表 L 的第 i 个元素的位置上。设计算法步骤如下：

①进行 i 值合法性判断；

②若空间已满，对空间进行扩展，并修正扩展后的顺序表各参数；

③从第 n 个元素逆序到第 i 个元素，每个元素后移一个位置；

④将新元素 x 存放在第 i 个元素的位置；

⑤修正插入后顺序表的参数。

（3）删除算法

删除顺序表 L 中第 i 个元素，被删元素通过参数 e 带回。设计算法步骤如下：

①进行 i 值合法性判断；

②保存被删元素到参数 e 中；

③从第 i+1 个元素到第 n 个元素，依次将每个元素前移一个位置；

④修正删除后顺序表的参数。

5．参考程序

```
#define OVERFLOW -1
#define OK 1
#define ERROR 0
#define MAXSIZE 6          //空间初始分配量，为了检验空间扩展，初值取小一点
#define INCREMENT 10       //空间分配增量
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>

typedef int Status;
typedef int ElemType;
typedef struct {
```

```
    ElemType *elem;              //存储空间基址
    int length;                  //当前顺序表中实际元素的个数
    int listsize;                //当前分配的存储容量
} SqList;

Status InitList(SqList &L)       // 初始化顺序表
{
    L.elem=(ElemType *)malloc(MAXSIZE*sizeof(ElemType));
    if( __1_____________________ )   exit(OVERFLOW);
    L.length=0;
    __2_____________________________;
    }
    return OK;
}

Status   InsertList (SqList &L, int i, ElemType e) {
//将新元素 e 插入到顺序表 L 的第 i 个位置上
    if (__3_____________________ )   return   ERROR;
    ElemType   *p;
    int   j;
    if(__4_____________________ ) {        //若顺序表已满，则需扩充空间
        p=(ElemType *)realloc(L.elem, (L.listsize+INCREMENT)*sizeof(ElemType));
        if (!p) exit (OVERFLOW);
        L.elem=p;
        L.listsize += INCREMENT;
    }
    for(j = L.length-1; j>=i-1; --j) L.elem[j+1]=L.elem[j];
    __5_____________________ ;            //插入新元素
    L.length++;
    return OK;
}// InsertList

void PrintList(SqList L)                 // 输出顺序表元素
{
    int   i;
    for(i=0;__6_____________________; i++)
        printf("%d   ", L.elem[i]);
    printf("\n");
}
```

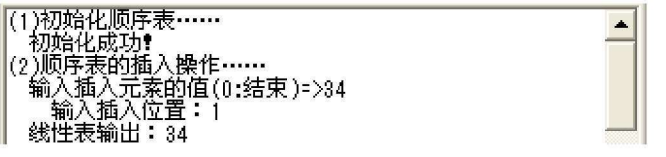
```
int SearchList(SqList L, ElemType e)
// 在顺序表 L 中查找值为 e 的第一个元素，查找成功返回元素的位置，失败返回-1
{
    int i;
    for(i=0; i<L.length; i++)
        if(___7_____)          // 找到相同的元素，返回位置
            return i;
    return -1;
}

int Del_List1(SqList &L, ElemType e)
// 在顺序表 L 中删除值为 e 的第一个元素，删除成功返回元素位置，失败则返回-1
{
    int i, j;
    for(i=0; i<L.length; i++)
        if(L.elem[i]==e)                    // 找到相同的元素
            break;
    if(i<L.length){
        for(j=i; j<L.length -1; j++)        // 删除
            ___8_____ ;
        L.length--;
        return i;
    }
    return -1;
}

Status    Del_List2(SqList &L, int i, ElemType &e)
// 在顺序表 L 中删除第 i 个元素，被删元素用参数 e 带回
{
    if (___9_____) return    ERROR;
    int    j;
    e=L.elem[i-1];
    for(j=i; j<L.length; j++)
        ___10_____;
    --L.length;
    return OK;
}
```

```
void main()
{
    SqList   LL;
    ElemType x;
    int r,i;
    printf("(1)初始化顺序表……\n");
    if( !InitList(LL) )   return;
    printf("   初始化成功！\n");
    printf("(2)顺序表的插入操作……\n");
    while(1)
    {
        printf("   输入插入元素的值(0:结束)=>");
        scanf("%d", &x);
        if(__11_____)
            break;
        printf("      输入插入位置：");
        scanf("%d", &r);
        ___12_____ ;
        printf("   线性表输出：");
        ___13_____ ;
    }
    printf("(3)顺序表上的查找操作……\n");
    while(1)          //在顺序表中查找指定值的元素，输出该元素所在位置
    {
    printf("   输入查找元素的值(0:结束)=>");
        scanf("%d", &x);
        if(x==0)
            break;
        r=___14_____;
        if(r<0)
            printf("      没找到！\n");
      else
            printf("   有符合条件的元素，位置为：%d\n", r+1);
    }
    printf("(4)顺序表中指定元素值的删除操作……\n");
    while(1)            //在顺序表中删除指定值的元素
    {
    printf("   输入删除元素的值(0:结束)=>");
        scanf("%d", &x);
```

```
        if(x==0)
            break;
        r=___15_____________________;
        if(r<0)
            printf("   没找到\n");
        else {
            printf("   删除成功！被删元素的位置是：%d\n   线性表输出：", r+1);
            PrintList(LL);
        }
    }
    printf("(5)顺序表中指定元素位置的删除操作……\n");
    while(1)
    {
        printf("   输入删除元素的位置(0:结束)=>");
        scanf("%d", &r);
        if(r==0)
            break;
        if( _16_____________________)
            printf("   位置越界!\n");
        else {
            printf("   线性表输出：");
            PrintList(LL);
        }
    }
}
```

6．测试结果（如图 1.1.1 所示）

从图 1.1.1 的测试结果可以看出：（1）当插入第 7 个元素 45 时，能正确显示 7 个元素值，说明插入成功。但顺序表在初始化时，只申请了 6 个元素的空间，表明插入第 7 个元素时空间进行了扩展。（2）在查找操作中，只能找到第一个满足条件的元素的位置。（3）在删除指定值的操作中，也只能删除第一个值相符的元素。



… …

```
  输入插入元素的值(0:结束)=>36
    输入插入位置：6
  线性表输出：34  12  45  64  28  36
  输入插入元素的值(0:结束)=>45
    输入插入位置：7
  线性表输出：34  12  45  64  28  36  45
  输入插入元素的值(0:结束)=>0
(3)顺序表上的查找操作……
  输入查找元素的值(0:结束)=>45
  有符合条件的元素，位置为：3
  输入查找元素的值(0:结束)=>0
(4)顺序表中指定元素值的删除操作……
  输入删除元素的值(0:结束)=>45
  删除成功！被删元素的位置是：3
  线性表输出：34  12  64  28  36  45
  输入删除元素的值(0:结束)=>0
(5)顺序表中指定元素位置的删除操作……
  输入删除元素的位置(0:结束)=>1
  线性表输出：12  64  28  36  45
  输入删除元素的位置(0:结束)=>6
   位置越界
  输入删除元素的位置(0:结束)=>0
Press any key to continue
```

图 1.1.1　顺序表的基本操作测试结果

7．课外实验

改用静态数组方式实现上述算法。

实验题 1.2　顺序表的基本运算

1．问题描述

创建一个顺序表，编程实现顺序表的下列基本运算：

（1）在顺序表的第 i 个位置上插入 m 个新元素；

（2）删除顺序表中元素值在 x 到 y 之间的所有元素。

2．基本要求

（1）采用动态分配方式设计顺序表的存储结构；

（2）插入算法必须考虑存储空间的动态扩充；插入和删除两种算法均必须满足时间复杂度为 O(n)，空间复杂度为 O(1)；

（3）根据插入和删除的算法分析，请在参考程序中的下划线处填上适当的子句，完善参考程序；

（4）设计测试数据，上机调试、测试参考程序，保存和打印测试结果，对测试结果进行分析，重点分析两种算法的性能是否达到第（2）条的要求。

3．算法分析

（1）插入算法

① 判断 i、m 的合法性；

② 判断 L.length+m 是否大于 L.listsize，是则扩充空间；

③ 将 L 中的第 n 到第 i 个元素逆序向后移动 m 个位置；

④ 在位置 i 处插入 m 个新元素（可以是已知数组，也可键盘输入）；

⑤ 调整 L 的参数，算法结束。

（2）删除算法

在线性表中设置两个初值为 0 的下标变量 i 和 j，其中，i 为比较元素的下标，j 为赋值元素的下标。依次取线性表中下标为 i 的元素与 x 和 y 比较，假若是 x 到 y 之外的元素，则赋值给下标为 j 的元素。这种算法比删除一个元素后立即移动其后面的元素的效率高得多。

4．参考程序

```
#define MAXSIZE 10              //空间初始分配量
#define OK 1
#define ERROR 0
#define OVERFLOW -1
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>

typedef int Status;
typedef int ElemType;          //将元素类型定义为整型
typedef struct {
    ElemType *elem;            //存储空间基址
    int length;                //当前顺序表中实际元素的个数
    int listsize;              //当前分配的存储容量
} SqList;

Status  CreatList(SqList &L) {        // 建立有 n 个元素的顺序表
    int i;
    do {                              // 若表的长度输入不合法，则循环输入
        printf("请输入元素个数：");
        scanf("%d", &L.length );
    } while(__1_____________________________);
    L.elem=(ElemType *)malloc(MAXSIZE*sizeof(ElemType));
    if(!L.elem) return ERROR;
    for(i=0; i<L.length; i++)   {         // 依次输入 n 个元素
        printf("请输入第%d 个元素值=>", i+1);
        scanf("%d", __2_____________________);
    }
    return OK;
}
```