数据结构实验指导书

卢玲，陈媛，何波，刘洁，苟光磊，刘恒洋编著

目　　录

1 补充知识

1.1 类 C 语言与标准 C 的转换要点

教材中的部分算法采用类 C 语言进行描述，在实验时，注意将类 C 语言的伪代码转换成 C 语言的源代码。

1．预定义常量和类型的问题

在类 C 语言的算法中出现了下列常量，验证时就需在源程序中定义：

```
#define TRUE    1
#define FALSE    0
#define OK    1
#define ERROR    0
#define INFEASIBLE    -1
#define OVERFLOW    -2
#define int    Status
```

2．算法描述的变量问题

在类 C 语言算法（函数）中，辅助变量没有作变量说明，但在验证源程序的函数定义时，需对所有使用的变量（除函数参数外）作说明。

【例 10-1】 如下是顺序表的插入算法，采用类 C 语言算法描述：

```
Status ListInsert_Sq(SqList &L, int pos, ElemType e)
{
  //在顺序线性表 L 的第 pos 个元素之前插入新的元素 e，pos 为位序
  //pos 的合法值为 1≤pos≤Listlength_Sq(L)+1
  if (pos < 1 || pos > L.length+1) return ERROR;      //插入位置不合法
  if (L.length >= L.listsize)  //当前存储空间已满，增加分配
  {
    newbase=
        (ElemType*)realloc(L.elem,(L.listsize+LISTINCREMENT)*sizeof (ElemType));
    if (!newbase) exit(OVERFLOW);       //存储分配失败
    L.elem = newbase;                   //新基址
    L.listsize += LISTINCREMENT;       //增加存储容量
  }
  q=&(L.elem[pos-1]);      //q 指示插入位置
  for (p = &(L.elem[L.length-1]); p >= q; --p)
      *(p+1) = *p;           //插入位置及之后的元素右移
  *q=e;                     //插入 e
   ++L.length;              //表长增 1
   return OK;
```

```
}
```

伪代码转换成的 C 语言源代码如下：

```
int ListInsert_Sq(SqList *L, int pos, ElemType e)
{
    ElemType *newbase,*p,*q;
    if (pos < 1 || pos > L->length+1) return (-1);
    if (L->length >= L->listsize)
    {
        newbase=
          (ElemType*)realloc(L->elem,(L->listsize+LISTINCREMENT)*sizeof (ElemType));
        if (!newbase) exit(-1);
        L->elem = newbase;
        L->listsize += LISTINCREMENT;
    }
    q =&((L->elem)[pos-1]);
    for (p=&((L->elem)[L->length-1]);p>=q;--p)
       *(p+1)=*p;
    *q = e;
    ++L->length;
    return (1);
}
```

3.算法描述的参数问题

在类 C 语言算法（函数）中，在形参定义时，以“&”打头的参数作为引用参数，即在函数调用后发生改变的量，但在验证源程序的函数定义时，需将引用参数转换为指针类型，在函数定义内，把引用参数的使用转换为引用参数对象的使用。

【例 10-2】 顺序表的插入算法，用类 C 语言描述如下：

```
Status ListInsert_Sq(SqList &L, int pos, ElemType e)
{
  if (pos < 1 || pos > L.length+1) return ERROR;
  if (L.length >= L.listsize)
  {
   newbase =
        (ElemType *)realloc(L.elem,(L.listsize+LISTINCREMENT)*sizeof (ElemType));
   if (!newbase) exit(OVERFLOW);
   L.elem = newbase;
   L.listsize += LISTINCREMENT;
  }
  q = &(L.elem[pos-1]);
```

```
    for (p = &(L.elem[L.length-1]); p >= q; --p) *(p+1) = *p;
    *q = e;
    ++L.length;
    return OK;
}
```

伪代码转换成的 C 语言源代码如下：

```
int ListInsert_Sq(SqList *L, int pos, ElemType e)
{
     ElemType *newbase,*p,*q;
     if (pos < 1 || pos > L->length+1) return (-1);
     if (L->length >= L->listsize)
    {
         newbase=
            (ElemType*)realloc(L->elem,(L->listsize+LISTINCREMENT)*sizeof (ElemType));
        if (!newbase) exit(-1);
        L->elem = newbase;
        L->listsize += LISTINCREMENT;
    }
    q =&((L->elem)[pos-1]);
    for (p=&((L->elem)[L->length-1]);p>=q;--p)
        *(p+1)=*p;
    *q = e;
    ++L->length;
    return (1);
}
```

4.由算法描述转换为程序代码

对本书的算法，均以自定义的 C 语言函数形式给出。要验证该算法，还需编写完整的源程序。在完整的源代码中，通过调用自定义的函数，实现算法的功能。

一个完整的源程序的结构如下。注意其中的算法表现为函数，算法的执行则表现为函数调用。

```
文件包含预处理
符号常量的定义
类型定义        //确定数据结构
返回类型 自定义函数名（形式参数表） //自定义的函数，即算法
{
      …        //算法实现的相关代码
}
void main()
```

```
{
    变量定义；          //定义处理对象
    建立对象；          //根据存储类型，给变量赋值，以确定具体的处理对象
    调用自定义函数；  //调用函数，实现算法的功能
    打印输出；          //打印结果
}
```

一个完整的由算法转换成的 C 源程序代码如例 10-3 所示。

【例 10-3】有一个单链表的就地逆置算法，现编写一个完整的源程序验证该算法。

【分析】本例中包含三个单链表操作的算法，分别是：create（创建）、disp（显示）、invert（逆置）。这三个函数分别是一个完整的算法，具有各自独立的功能。在 main 函数中，通过调用这三个函数，执行其中的操作，从而实现了对单链表的逆置及输出操作。

```
typedef int elemtype;     //定义元素类型
typedef struct linknode
{
elemtype data;
struct linknode *next;
} nodetype;              //定义结点类型，确定线性表的链式存储结构
nodetype *create()    //建立一个不带头结点的单链表，通过函数的值返回头指针
{
    elemtype d;    //变量定义
    nodetype *head=NULL,*s,*t;   //h 为头指针，t 为表尾结点的指针，s 指向新结点
    int i=1;          //i 记录结点的位序号
    printf("建立一个单链表\n");
    while (1)     //循环体完成新结点的插入，以实现链表的建立
   {
       printf("输入第%d 节点 data 域值：",i);
       scanf("%d",&d);
       if (d==0)  break;                //以 0 表示输入结束
       if(i==1)                         //建立第一个结点
      {
           head=(nodetype *)malloc(sizeof(nodetype));
           head->data=d;head->next=NULL;t=head;
      }
      else
      {
           s=(nodetype *)malloc(sizeof(nodetype));
           s->data=d; s->next=NULL; t->next=s;
           t=s;      //t 始终指向生成的单链表的最后一个结点
      }
```

```
        i++;
     }
     return head;    //返回头指针
}
void disp(nodetype *head)    //遍历显示以 h 为头指针的单链表
{
    nodetype *p=head;       //p 指向正在处理的结点
    printf("输出一个单链表：\n");
    if (p==NULL)
        printf("空表");
    while (p!=NULL)
    {
        printf("%d ",p->data);
        p=p->next;
    }
    printf("\n");
}
nodetype *invert(nodetype *head)    //就地逆置单链表
{
    nodetype *p, *q, *r;
    if (!head||!(head->next))
    {
        printf("逆置的单链表至少有 2 个节点\n");
        return NULL;
    }
     else   //就地逆置
     {
        p=head; q=p->next;
        while (q!=NULL)
        {
            r=q->next;
            q->next=p;
            p=q; q=r;
        }
        head->next=NULL;
        head=p;
        return head;
     }
}
```

```
void main()  //主函数
{
    nodetype *head;  //定义变量 head，以表示处理的单链表头指针
    head=create();   //调用算法 creat，建立单链表
    disp(head);      //调用算法 disp，显示逆置前的单链表
    head=invert(head); //调用逆置函数 invert，实现逆置功能
    disp(head);      //调用算法 disp，显示逆置后的单链表
}
```

1.2 实验内容

实验 1 C 语言的数组

1 实验目的

数组是 C 语言的一种固有数据类型，在各类问题中具有非常广泛的应用。例如记录某班全部同学的 C 语言成绩，或者记录灰度图像的直方图等。数组本质上是一个包含若干分量的数据集合，它的所有分量具有相同的数据类型，且在内存中占据连续的存储空间。

对数组的熟练操作和灵活运用，是学习和掌握绝大多数复杂数据结构的基础。本实验的目的是复习 C 语言数组的知识，为熟练编写具有复杂数据结构的程序做好准备。

2 基础练习

【练习一】简单的整型数组操作：构造一个包含 n 个元素的一维整型随机数组（n≤100），其中无重复元素。完成表 10-1 的操作。

表 10-1 整型数组的基本操作
⑴ 使用下标法输出数组元素值
⑵ 使用指针法输出数组元素值
⑶ 对数组中所有偶数求和
⑷ 对数组中所有奇数求和
⑸ 将数组中所有数扩大 k 倍
⑹ 把数组中偶数位置的元素加起来
⑺ 找出数组中的最大值
⑻ 找出数组中的最小值
⑼ 在数组中搜索指定元素，看其是否存在

【练习二】简单的字符数组操作：构造一个长度不超过 n 的一维字符数组（n≤100），完成表 10-2 的操作。

表 10-2 字符数组的基本操作
⑴ 使用下标法输出该字符串
⑵ 使用指针法输出该字符串
⑶ 统计字符串中小写字母的个数
⑷ 统计字符串中原音字母(小写)的个数(重复字母不重复计数)
⑸ 找出字符串中的最大值
⑹ 找出字符串中的最小值

【练习三】简单的数组操作：构造一个 n×n 的（2≤n,m≤100）矩阵，用于存储整型数据。完成表 10-3 的操作。

表 10-3 n×n 矩阵的基本操作
⑴ 使用下标法输出该矩阵

⑵ 使用指针法输出该矩阵
⑶ 统计矩阵中非 0 元的个数
⑷ 求矩阵中所有元素的和
⑸ 将矩阵转置

3 进阶练习

【练习一】构造一个包含 n 个元素的一维随机整型数组（n≤100）。完成表 10-4 的操作：

表 10-4 一维整型数组的其它操作
⑴ 删除数组中指定值的元素
⑵ 删除数组中指定位置的元素
⑶ 删除数组中值相同的元素
⑷ 在数组中指定的位置添加元素
⑸ 对元素进行排序
⑹ 向有序的数组中添加元素，并保持有序性
⑺ 将数组元素反序

【练习二】构造一个长度不超过 n 的一维字符数组（n≤100），完成表 10-5 的操作：

表 10-5 一维字符数组的其它操作
⑴ 删除字符串中指定字符
⑵ 删除字符串中指定位置的元素
⑶ 删除字符串中所有的原音字母
⑷ 在字符串中指定的位置添加一个字符

【练习三】构造两个 n×n 的（2≤n,m≤100）矩阵 A 和 B，用于存储整型数据。完成表 10-6 的操作。

表 10-6 n×n 矩阵的其它操作
⑴ 判断矩阵 A 是否是对称矩阵
⑵ 判断矩阵 B 是否是上三角矩阵
⑶ 将矩阵 A 的下三角元素存储到一个一维数组中
⑷ 将矩阵 A 的上三角元素存储到一个一维数组中
⑸ 实现 A+B

4 扩展练习

【练习一】稀疏矩阵是这样的矩阵，它只有 1/3 左右的元素非 0，而其余元素都是 0。

【问题】编写程序，判断任意整型矩阵是否是稀疏矩阵。

【练习二】加密技术是把消息（明文）变换成一种伪装的形式（密文）进行秘密通信的一种方法，除收件人之外，任何人看了密文也不能翻译成明文。把明文变换成密文称为加密，把密文转换成明文称为解密。Twisting(扭曲)是一个简单的加密方法，它需要发送者和接收者都共同认可的加密关键字k，k是一个正整数。

Twisting方法使用4个数组：plaintext和ciphertext是字符数组，plaincode和ciphercode是整型数组。所有数组的长度为n，这是对信息加密的长度。所有数组初始时为空，下标从0到n-1。消息只包含小写字母，句号和下画线（代表空格）。

消息存储在数组plaintext中。给定关键字k，加密方法如下：首先把plaintext的字母转换成数字编码存放到数组plaincode中，转换规则：‘__’=0，‘a’=1，‘b’=2，…‘z’=26，‘.’=27。然后将存放在数组plaincode中的数字编码按下列公式转换成加密代码存放到数组ciphercode中：i从0到n-1。

ciphercode[i] = (plaincode[k*i mod n] - i) mod 28.

这里 mod 是模运算。最后，把存放在 ciphercode 中的数字编码按上述方法转换成密文存放到数组 ciphertext 中。

【问题】编写程序，实现消息的untwist（解密），即给定关键字k，将密文恢复至原来的明文。例如，关键字是5，密文是’cs.’，程序必须输出明文’cat’。

注意：密文至少包括一个字符，最多70个字符。关键字k是一个正整数，不超过300。可以假定解密消息的结果是唯一的。

表10-7是几组测试数据。

表 10-7 几组测试数据

K(输入数据)	密文(输入数据)	明文(输出数据)
5	cs.	cat
101	thqqxw.lui.qswer	this_is_a_secret
3	b_ylxmhzjsys.virpbkr	beware._dogs_barking

实验 2 C 语言的指针和结构体

1 实验目的

本实验的目的是复习 C 语言指针和结构体的有关知识，为熟练编写具有复杂数据结构的程序做好准备。

2 基础练习

【练习一】编写函数 fun：判断两个指针 a 和 b 所指存储单元的值的符号是否相同；若相同函数返回 1，否则返回 0。这两个存储单元中的值都不为 0。在主函数中输入 2 个整数、调用函数 fun，输出结果。

【函数原型】int fun(int *a, int *b)；

【练习二】编写函数 sum：求数组 a 中所有奇数之和和所有偶数之和。形参 n 给出数组中数据的个数；利用指针 odd 和 even 分别返回奇数之和和偶数之和。在主函数中输入 n 个整数，调用函数 sum 进行求和，输出计算结果。

【函数原型】void sum(int *a, int n, int *odd, int *even)；

【练习三】编写函数 dcopy：将数组 a 中的偶数复制到数组 b 中。在主函数中输入 n 个整数、调用函数 dcopy 进行复制，输出复制后的数组 b。

【函数原型】int dcopy(int *a, int *b, int n)；

【练习四】编写函数 exchange：把数组 a 中的最大数和最小数交换（设 a 中无重复元素）。在主函数中输入 n 个整数，调用函数 exchange 进行交换，输出运行结果。

【函数原型】int exchange(int *a, int n)；

【练习五】有 n（n≤40）个学生，每个学生的数据包括学号、姓名、3 门课程的成绩。编写函数：

⑴ 输出 n 个学生的所有信息；

⑵ 输出每门课程的平均成绩；

⑶ 输出最高分的学生的信息。

3 进阶练习

【练习一】设计一个可进行复数运算的演示程序。要求实现下列六种基本运算：

⑴ 由输入的实部和虚部生成一个复数；

⑵ 两个复数求和；

⑶ 两个复数求差；

⑷ 两个复数求积；

⑸ 从已知复数中分离出实部；

⑹ 从已知复数中分离出虚部。

运算结果以相应的复数或实数的表示形式显示。

【提示】可定义结构类型，用于表示一个复数。其中包含两个分量，分别是复数的实部和虚部的系数。

【练习二】已知二维空间 XY 平面上的三个点，以它们为顶点构成一个三角形（面积 S≥0）。

定义结构类型，用于描述一个三角形。并进行下列运算：

⑴ 判断三角形是否是等边三角形；

⑵ 判断三角形是否是等腰三角形；

⑶ 检查三角形是否是直角三角形；

⑷ 求三角形的面积；

⑸ 检查某个点是否位于三角形内部。

【提示】可定义结构类型 point，用于描述 XY 平面的一个点(x,y)。在定义 point 类型的基础上，定义结构类型，用于描述 XY 平面上的一个三角形（包括三个点）。

4 扩展练习

【练习一】使用结构和文件，编写一个电话号码簿模拟程序。电话号码簿可以永久地存储在文件中，其中包括朋友的姓名、地址、电话号码、电子邮箱等等。可以对其中的记录进行增加、删除、查询、修改等操作。该号码簿的容量为 1500 条记录。

【练习二】在三维空间 XYZ 中存在 100 个粒子，可以使用移动的点来建立这些粒子的模型，忽略其大小。任何粒子的 x 坐标按 sin(i)移动，y 坐标按 cos(i)移动，z 坐标按 tan(i)移动，其中 i 是粒子的编号。编写一个描绘粒子路径的程序。

【提示】建立结构类型 point，用于描述一个粒子的坐标(x,y,z)。

实验 3 线性表

1 实验目的

熟练掌握线性表的基本操作，包括：创建、插入、删除、查找、输出、求长度、合并等运算，以及各类操作在顺序存储结构和链式存储结构上的实现。

2 基础练习

【练习一】顺序表基础操作：构造一个顺序表，其最大长度为 n（n≤1024）。每个元素中记录着一个整型的键值 key(设键值唯一)。编写函数，完成表 10-7 的操作：

表 10-7 线性表的基本操作

⑴ 初始化线性表
⑵ 输出线性表
⑶ 取表中的第 i 个元素的键值
⑷ 从表中删除指定位置的元素
⑸ 从表中删除指定键值的元素
⑹ 向表的头部添加键值为 key 的元素
⑺ 向表的尾部添加键值为 key 的元素
⑻ 向表中指定的位置 pos 处添加键值为 key 的元素
⑼ 在表中中搜索键值为 key 的元素，看其是否存在

【练习二】单链表基础操作：构造一个带头结点的单链表。其每个结点中记录着一个字符型的键值 key(设键值唯一)。编写函数，完成表 10-7 的操作。

【练习三】单循环链表基础操作：构造一个不带头结点的单循环链表。其每个结点中记录着一个整型的键值 key(设键值唯一)。编写函数，完成表 10-7 的操作。

【练习四】双链表基础操作：构造一个带头结点的双向链表。其每个结点中记录着一个整型的键值 key(设键值唯一)。编写函数，完成表 10-7 的操作。

【练习五】双向循环链表：构造一个带头结点的双向循环链表。其每个结点中记录着一个整型的键值 key(设键值唯一)。编写函数，完成表 10-7 的操作。

3 进阶练习

【练习一】编写函数，从一个顺序表 A 中删除元素值在 x 和 y(x≤y)之间的所有元素，要求以较高的效率来实现。

【练习二】编写函数，将一个顺序表 A（有 n 个元素且任何元素均不为 0），分拆成两个顺序表 B 和 C。使 A 中大于 0 的元素存放在 B 中，小于 0 的元素存放在 C 中。

【练习三】编写函数，用不多于 3n/2 的平均比较次数，在一个顺序表 A 中找出最大和最小值的元素。

【练习四】已知一个单链表如图 10-1 所示，编写一个函数将该单链表复制一个拷贝。

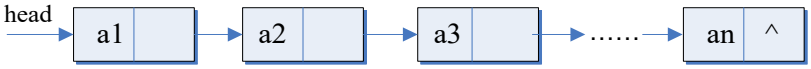


图 10-1 一个单链表

【练习五】如下类型定义：

```
typedef struct node
{    int exp;          //指数
     float coef;       //系数
     struct node *next;
}polynode;
```

【要求】用链式存储结构实现：生成两个多项式 PA 和 PB，求 PA 和 PB 之和，输出“和多项式”。

4 扩展练习

【练习一】约瑟夫生者死者问题。据说著名犹太历史学家 Josephus 有过以下的故事：在罗马人占领乔塔帕特后，39 个犹太人与 Josephus 及他的朋友躲到一个洞中，39 个犹太人决定宁愿死也不要被敌人抓到，于是决定了一个自杀方式：41 个人排成一个圆圈，由第 1 个人开始报数，每报数到第 3 人该人就必须自杀，然后再由下一个重新报数，直到所有人都自杀身亡为止。然而 Josephus 和他的朋友并不想遵从，Josephus 要他的朋友先假装遵从，他将朋友与自己安排在第 16 个与第 31 个位置，于是逃过了这场死亡游戏。这就是著名的约瑟夫生者死者问题。

17 世纪的法国数学家加斯帕在《数目的游戏问题》中也讲了这样一个故事：15 个教徒和 15 个非教徒在深海上遇险，必须将一半的人投入海中，其余的人才能幸免于难，于是想了一个办法：30 个人围成一圆圈，从第一个人开始依次报数，每数到第九个人就将他扔入大海，如此循环进行直到仅余 15 个人为止。

【问题】怎样的安排才能使每次投入大海的都是非教徒？请编程解决这一 n(1≤n≤30)个人的跳海问题。要求分别用两种线性表的存储结构来解决。

【提示】在使用链式存储结构时，可构造具有 30 个结点的单循环链表。

【练习二】数字信号（如图 10-2 所示），可以用两个属性来描述：信号在时间轴上的起点，以及任何特定时刻的振幅。因此，可以定义如下结构，在离散的时间轴上表示数字信号。

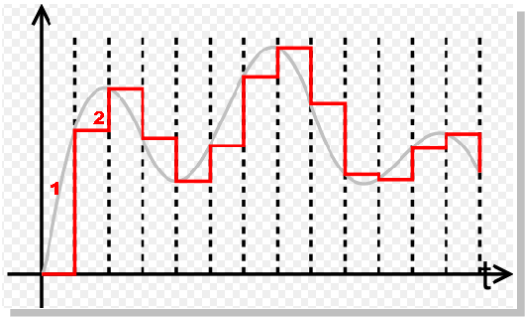


图 10-2 数字信号示意图

```
typedef struct Signal
{
     int time;           //时间
    int amplitude;      //振幅
```

```
    struct Signal *next;
}Signal;
```

建立具有上述结点结构的链表，该链表就可以用于描述图 10-2 中所示的数字信号了。

【问题】建立链式存储结构，编写程序，完成下列操作：

⑴ 记录若干数字信号结点的信息；

⑵ 查询信号中给定振幅的位置；

⑶ 在信号的末尾添加一个新的值；

⑷ 在信号的前部添加一个新的值；

⑸ 显示数字信号；

⑹ 得到信号中特定振幅的频率。

实验 4 栈和队列

1 实验目的

⑴ 深入了解栈和队列的特性，以便灵活应用。

⑵ 熟练掌握栈的两种构造方法、基于栈的各种操作和应用。

⑶ 熟练掌握队列的两种构造方法、基于队列的各种操作和应用。

2 基础练习

【练习一】如下结构，它表示一个能够保存 1024 个整数的整型顺序栈。

```
#define MAX 1024
typedef struct stack
{
    int data[MAX];
    int top;
}STACK;
```

编写函数，完成表 10-8 的操作：

表 10-8 栈的基本操作

⑴ 初始化栈
⑵ 显示栈顶元素
⑶ 将一个元素入栈
⑷ 从栈中弹出一个元素
⑸ 判栈是否为空
⑹ 判栈是否为满

【练习二】如下结点结构，试用该结构构造一个链栈。

```
typedef struct link
{
    int info;
   link *next;
}Link;
```

编写函数，完成如表 10-8 所示操作。

【注意】链栈是否要判栈满？

【练习三】如下结构，它是一个能保存 1023 个整数的整型循环队列。

```
#define MAX 1024
typedef struct queue
{
    int front;
    int rear;
    int data[MAX];
```

```
}QUEUE;
```

编写函数，完成表 10-9 的操作。

表 10-9 队列的基本操作
⑴ 初始化队列
⑵ 将一个元素入队
⑶ 将一个元素出队
⑷ 判队列是否为空
⑸ 判队列是否为满

【练习四】如下结点结构，试用该结构构造一个链队列。

```
typedef struct node
{
    int data;
    struct node *next;
}Node;
```

编写函数，完成如表 10-9 所示操作。

【注意】链队列是否要判队列满？

3 进阶练习

【练习一】单链表中有 n 个结点，每个结点中包含一个整型键值，编写算法将其逆置。

【提示】可利用【基础练习】中定义的栈结构及基本操作来实现。

【练习二】把相同的词汇或句子，在下文中调换位置或颠倒过来，产生首尾回环的情景，叫做回文，也叫回环。如清代黄伯权的《茶壶回文诗》就是一首“通体回文”诗。

落雪飞芳树，幽红雨淡霞。薄月迷香雾，流风舞艳花。

其诗又可回读为：

花艳舞风流，雾香迷月薄。霞淡雨红幽，树芳飞雪落。

在程序设计中，通常将称正读和反读都相同的字符序列称为“回文”。

如：“abba”、“abccba”、12321、123321 是回文；

“abcdef”、“abab”不是回文。

【问题】编程，检查给定的字符串是否是回文，忽略空白和其它符号。可使用下述输入测试你的程序，”Madam I am Adam”，”was it a cat I saw”和”Level”。对其中的字母不区分大小写。

【提示】本题可使用栈结构。由于字符串长未知，可使用链栈作为存储结构。

【练习三】编译器使用了栈结构。例如，在检查程序语法时，用栈结构检查括号匹配的问题。具体操作如下：

⑴ 从源代码文件中逐个读入字符；

⑵ 每读入一个{时，将一个对象（任何对象，具体是什么并不重要）压入栈中；

⑶ 每读入一个}时，将一个对象（任何对象，具体是什么并不重要）从栈中弹出；

当读入一个}时栈为空，那么一定缺少了{；

⑷ 如果读到文件末尾栈还不为空，那么一定缺少了}。

【问题】编写程序，检查源程序中的括号{和}是否匹配。

【提示】可利用【基础练习】中定义的栈结构及基本操作来实现。

【练习四】到医院看病时，患者需排队等候，排队过程中主要重复两件事：

⑴ 病人到达诊室时，将病历交给护士，排到等候队列中候诊。

⑵ 护士从等候队列中取出下一个患者的病历，该患者进入诊室就诊。在排队时按照”先到先服务”的原则。

【问题】设计一个算法模拟病人等候就诊的过程。

【提示】采用一个队列，有“病人到达”命令时即入队，有“护士让下一位患者就诊”命令时即出队。由于队列人数未知，建议采用链队列。

4 扩展练习

【练习一】如果一个数从左边读和从右边读一样，那么我们说这是一个回文数。例如，75457 是一个回文数。当然，这个数的特性还依赖于表示它的进制。如果用十进制表示 17，它不是回文数；但是如果用二进制（10001）表示它，它就是一个回文数。

【问题】编写程序，判断某数在二~十六进制下是否是回文数。程序输出的信息为：“Number i is palindrom in basis”，其中 i 是给定的数，接着输出进制，在该进制下数 i 是回文。如果这个数在二~十六进制下都不是回文数，程序输出：“Number i is not palindrom”。

如输入数据为：

17

19

输出数据为：

Number 17 is palindrom in basis 2 4 16

Number 19 is not a palindrom

【练习二】将中缀表达式转换成逆波兰式。

表达式计算是实现程序设计语言的基本问题之一。在计算机中进行算术表达式的计算可通过栈来实现。通常书写的算术表达式由操作数、运算符以及圆括号连接而成。为简便起见，在这里只讨论双目运算符。

算术表达式的两种表示如下：

⑴ **中缀表达式：**把双目运算符出现在两个操作数中间的表示叫做算术表达式的中缀表示，这种算术表达式称为中缀算术表达式或中缀表达式。如表达式 2+5*6 就是中缀表达式。

⑵ **后缀表达式：**中缀表达式的计算比较复杂。能否把中缀算术表达式转换成另一种形式的算术表达式，使计算简单化呢？波兰科学家卢卡谢维奇(Lukasiewicz)提出了算术表达式的另一种表示，即后缀表式，又称**逆波兰式。**

逆波兰式是把运算符放在两个运算对象的后面。采用后缀表示的算术表达式被称为后缀算术表达式或后缀表达式。在后缀表达式中，不存在括号，也不存在优先级的差别，计算过程完全按照运算符出现的先后次序进行，整个计算过程仅需一遍扫描便可完成，比中缀表达式的计算简单得多。例如，12!4!-!5!/就是一个后缀表达式。其中’!’字符表示操作数之间的空格，因减法运算符在前，除法运算符在后，所以应先做减法，后做除法；减法的两个操作数是它前面的 12 和 4，其中第一个数 12 是被减数，第二个数 4 是减数；除法的两个操作数

是它前面的 12 减 4 的差(即 8)和 5，其中 8 是被除数，5 是除数。

中缀算术表达式转换成对应的后缀算术表达式的规则是：

把每个运算符都移到它的两个运算对象的后面，然后删除掉所有的括号即可。

表 10-10 是一些中缀表达式与后缀表达式对应的例子：

表 10-10 中缀表达式与对应的后缀表达式

中缀表达式	后缀表达式
3/5+6	3!5!/!6!+
16-9*(4+3)	16!9!4!3!+!*!-
2*(x+y)/(1-x)	2!x!y!+!*!1!x!-!/
(25+x)*(a*(a+b)+b)	25!x!+!a!a!b!+!*!b!+!*

【问题】编程，将任意一个合法的中缀表达式转换成逆波兰式。假定表达式中的数字均为 1 位整数，运算符包括：+、−、*、/、(、)。

【提示】中缀表达式和得到的后缀表达式，均用字符串的形式存储。从左向右顺序扫描中缀表达式，根据读取到的是数字字符还是运算符，分别进行处理。

实验 5 串

1 实验目的

⑴ 掌握串的基本概念、存储方法及主要运算。

⑵ 将串的运算应用到文本编辑中。

2 基础练习

【练习一】分析以下代码的输出结果？

```
void demonstrate()
{
    Assign(s,'THIS IS A BOOK');
    Replace(s,SubStr(s,3,7),'ESE ARE');
    Assign(t,Concat(s,'S'));
    Assign(u,'XYXYXYXYXYXY');
    Assign(v,SubStr(u,6,3));
    Assign(w,'W');
    printf('t=%s v=%s   u=%s   %s',t,v,u,replace(u,v,w));
}
```

【练习二】采用定长顺序存储结构存储串。设计一个算法：将串中所有的字符倒过来重新排列。

【练习三】采用定长顺序存储结构存储串。编写一个函数 index(s1,s2)：判断 s2 是否是 s1 的子串。若是，返回其在主串中的位置；否则返回-1。

【练习四】如果第二个字符串以第一个字符串作为开始字符串，我们就说第一个字符串是第二个字符串的前缀。例如，Wonder 是 Wonderful 的前缀，原因在于 Wonderful 以 Wonder 作为开始字符串。

编程：判断某字符串是否是另一个字符串的前缀。

3 进阶练习

【练习一】什么是字符串的子序列？

有时，人们会将子序列误认为是子串，这是不正确的看法。它们两者存在一些细微的差别。如果第一个字符串中的字符从左到右依次出现在第二个字符串中，这样出现没有必要像子串那样连续出现，那么第一个字符串称为第二个字符串的子序列。例如，Wine 是字符串 **W**orld **i**s **n**ot **e**nough 的子序列。

【问题】编程：判断一个字符串是否是另一个字符串的子序列。

假定出现在两个字符串中的字符都是唯一的。

【练习二】假设以定长顺序存储结构表示串，设计一个算法：

void maxcommonstr(SString s, SString t, int &maxlen, int &pos1, int &pos2);

求串 s 和串 t 的一个最长公共子串，其中带出参数 maxlen 为最长公共子串的长度，pos1 和 pos2 分别为最长公共子串在串 s 和串 t 中的起始位置。

【练习三】采用定长顺序存储结构存储串。利用串的基本运算，编写一个算法：删除串 s1 中所有 s2 子串。

【练习四】采用定长顺序存储结构存储串。编写一个实现串通配符匹配的函数 pattern_index()，其中的通配符只有'?'，它可以和任一字符匹配成功。

例如，pattern_index("?re"，"there are")返回的结果是 2。

4 扩展练习

【练习一】实现堆存储结构上串的各种基本运算及其应用算法

【练习二】试写一算法，实现堆存储结构的串的插入操作：StrInsert（&S，pos，T）。

【练习三】S='s1s2…sn'是一个长为 N 的字符串，存放在一个数组中，编程序将 S 改造之后输出：

⑴ 将 S 的所有第偶数个字符按照其原来的下标从大到小的次序放在 S 的后半部分

⑵ 将 S 的所有第奇数个字符按照其原来的下标从小到大的次序放在 S 的前半部分

例如：S='ABCDEFGHIJKL'；

则改造后的 S 为'ACEGIKJHFDB'。

实验 6 数组和广义表

1 实验目的

⑴ 熟练掌握数组的存储表示和实现。

⑵ 熟悉广义表的存储结构的特性。

2 基础练习

【练习一】设数组 R[0..n-1]的 n 个元素中有多个 0 元素。设计一个算法，将 R 中所有的非 0 元素依次移动到 R 数组的前端。

【提示】用 i 指向不为 0 元素应放的下标，用 j 遍历 R，当 R[j]不为 0 时，在 i 与 j 不相同时将 R[i]与 R[j]交换。

【练习二】设计一个算法，将 A[0..n-1]中所有奇数移到偶数之前。要求不另增加存储空间，且时间复杂度为 O(n)。

【提示】利用 i 从左向右遍历，指向 A 左边的一个偶数；利用 j 从右向左遍历，指向 A 右边的一个奇数，然后将 A[i]与 A[j]交换。如此循环直到 i 大于等于 j。

【练习三】如果矩阵 A 中存在这样的一个元素 A[i,j]满足条件：A[i,j]是第 i 行中值最小的元素，且又是第 j 列中值最大的元素，则称之为该矩阵的一个马鞍点。

编写函数：计算出 m×n（1≤m,n≤20）的矩阵 A 的所有马鞍点。

3 进阶练习

【练习一】如下稀疏矩阵 A。编写程序：用三元组表存储该稀疏矩阵，并以阵列形式输出该稀疏矩阵。

$$\begin{bmatrix} 50 & 0 & 0 & 0 \\ 10 & 0 & 20 & 0 \\ 0 & 0 & 0 & 0 \\ -30 & 0 & -60 & 5 \end{bmatrix}$$

【练习二】以三元组表表示法表示稀疏矩阵，实现两个矩阵相加、相减和相乘的运算。运算结果的矩阵以阵列形式列出。

4 扩展练习

【练习一】编写一个函数删除广义表中所有值为 x 的元素。例如删除广义表((a, b), a, (d, a))中所有 a 的结果是广义表((b),(d))。

【练习二】使用黑色墨水在白纸上签名，其实就是一些黑点（像素）所构成的稀疏矩阵。如图 10-3 所示。

图 10-3 手写体签名

【问题】请创建一个稀疏点阵信息（用于描述文字或图形），并保存于磁盘文件中。

⑴ 读取磁盘文件的点阵信息到稀疏矩阵；稀疏矩阵采用三元组表示；

⑵ 用 3 种不同的算法实现稀疏矩阵转置，转置后的矩阵也采用三元组表示；

⑶ 以阵列的形式输出转置后的矩阵。

实验 7 树和二叉树

1 实验目的

⑴ 熟练掌握二叉树的概念及存储方法。

⑵ 熟练掌握二叉树的遍历算法及基于二叉树遍历的各类应用。

⑶ 掌握实现树的各种运算的算法。

2 基础练习

【练习一】如下程序建立了一棵二叉树并进行中序遍历，请填空完成该程序。

```
#include " stdio.h"
#include "malloc.h"
struct node
{
    char data;
    struct node   *lchild , *rchild;
} bnode;
typedef struct node * blink;
blink add(blink bt,char ch)
{
    if(bt==NULL)
    {
        bt=nalloc(sizeof(bnode));
        bt->data = ch;
        bt->lchild = bt->rchild =NULL;
    }
    else   if( ch < bt->data)
              bt->lchild = add(bt->lchid ,ch);
        else
              bt->rchild = add(bt->rchild,ch);
    return bt;
}
void inorder(blink bt)
{
     if(bt)
     {
          inorder(bt->_____);
          printf("%c",_____);
          inorder(bt->_____);
     }
```

```
}
void main()
{
    blink root = NULL;
    int i,n;
    char x;
    scanf（“%d”,&n);
    for(i=0；i<=n；i++)
    {
        x=getchar();
        root=add(root,x);
    }
    inorder(root);
    printf（“\n”);
}
```

若上述程序的输入为：ephqsbma

则程序运行输出为：_____

【练习二】设二叉树的每个结点中包含一个整型键值，如图 10-4 所示。编程，建立二叉树的二叉链式存储结构，并实现如表 10-11 所示操作：

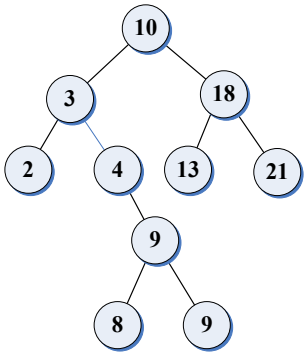


图 10-4 一棵二叉树

表 10-11 二叉树的基本操作
(1) 求二叉树的结点数和叶子数
(2) 按层次遍历二叉树
(3) 对二叉树进行先序遍历、中序遍历、后序遍历
(4) 求二叉树的深度
(5) 求二叉树中以元素值为 x 的结点为根的子树的深度

3 进阶练习

【练习一】一棵二叉树如图 10-5 所示。将其存于顺序表 sa 中，sa.elem[1..sa..last]含结点值。试编写算法：由此顺序存储结构建立该二叉树的二叉链表。

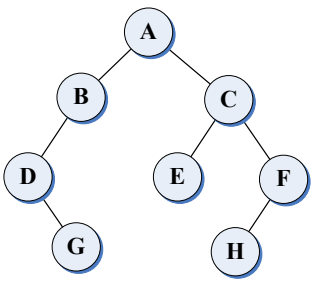


图 10-5 一棵二叉树

顺序表 sa:

A	B	C	D	⊙	E	F	⊙	G	⊙	⊙	H	⊙

【练习二】一棵树如图 10-6 所示。编写程序，以双亲表示法存储这棵树，并计算树的深度。

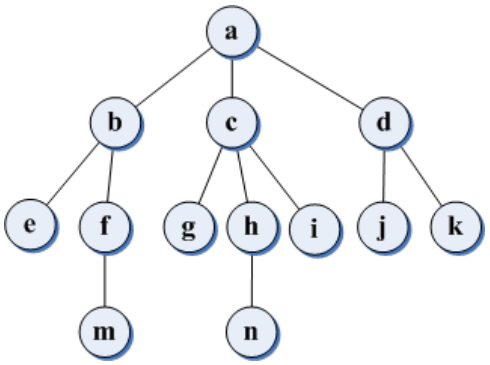


图 10-6 一棵树

4 扩展练习

【练习】编程，以孩子-兄弟链法存储一棵树，并实现如表 10-12 所示的操作：

表 10-12 树的基本操作
⑴ 统计该树的叶子数
⑵ 求该树的深度
⑶ 在树中查找值为 x 的结点
⑷ 对找到的某一结点，计算其所拥有的孩子的个数
⑸ 对找到的某一结点，找出其所有的兄弟
⑹ 对找到的某一结点，找出其所有的叔叔

实验 8 图

1 实验目的

⑴ 熟悉图的各种存储方法。

⑵ 掌握遍历图的递归和非递归的算法。

⑶ 理解图的有关算法。

2 基础练习

【练习一】如图 10-7 所示无向连通图，完成下面练习：

⑴ 编程，实现该图的邻接矩阵存储。

⑵ 编程，实现如下操作：

- InsertLine（G，e）：向图中添加一条边
- DeleteLine（G，e）：从图中删除一条边

【练习二】如图 10-8 所示有向图，完成下面练习：

⑴ 编程，实现该图的邻接表存储。

⑵ 编程，实现如下操作：

- InsertLine（G，e）：向图中添加一条边
- DeleteLine（G，e）：从图中删除一条边

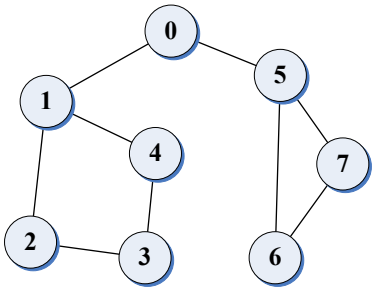


图 10-7 一个无向连通图

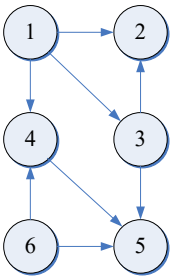


图 10-8 一个有向图

3 进阶练习

【练习一】在【基础练习】->【练习一】所建立的邻接矩阵上，从任意顶点开始，对邻接矩阵进行深度优先搜索，给出搜索序列。

【练习二】在【基础练习】->【练习二】所建立的邻接表上，从任意顶点开始，对邻接表进行广度优先搜索，给出搜索序列。

【提示】若有向图非强连通图，则一次遍历不能访问到所有的顶点。一次遍历结束后，还需从未被遍历到的顶点中重新选择顶点，再继续进行遍历，如此反复，直到所有顶点都被访问到为止。可以通过设置 visited[]数组，遍历时将 visited[i]置 1，表示顶点 i 被访问过。遍历后，若所有顶点 i 的 visited[i]均为 1，则该图是连通的；否则不连通。

【练习三】编写函数，实现将【基础练习】->【练习一】所建立的邻接矩阵存储转换成邻接表存储。

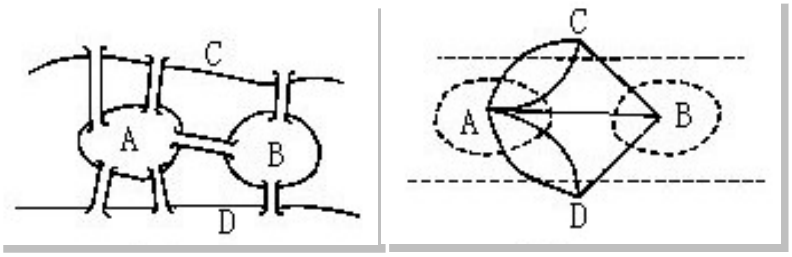
4 扩展练习

【练习一】编写程序，对图 10-8 所示有向图进行拓扑排序，要求用图的邻接表方式存储。

【练习二】1736 年瑞士数学家欧拉（Euler）发表了图论的第一篇论文“哥尼斯堡七桥问题”。在当时的哥尼斯堡城有一条横贯全市的普雷格尔河，河中的两个岛与两岸用七座桥连接起来，如图 10-9(a)所示。当时那里的居民热衷于一个问题：游人怎样才能不重复地走遍七座桥，最后又回到出发点呢？

瑞士著名数学家欧拉认为，人们关心的只是一次不重复地走遍这七座桥，而并不关心桥的长短和岛的大小，因此，岛和岸都可以看作一个点，而桥则可以看成是连接这些点的一条线。这样，一个实际问题就转化为一个几何图形（如图 10-9(b)）能否一笔画出的问题了。

欧拉用 A,B,C,D 这 4 个字母代替陆地，作为 4 个顶点，将联结两块陆地的桥用相应的线段表示，如图 10-9(b)所示。于是哥尼斯堡七桥问题就变成了图 10-9(b)中，是否存在经过每条边一次且仅一次，经过所有的顶点的回路问题了。欧拉在论文中指出，这样的回路是不存在的。



(a) (b)

图 10-9 哥尼斯堡七桥

【问题】设图 G 的一个回路，若它恰通过 G 中每条边一次，则称该回路为欧拉回路。存在欧拉回路的图就是欧拉图（**Euler Graph**）。编程，判断图 10-6(a)中所示的”哥尼斯堡七桥”是否是一个欧拉图。

【提示】**如果图中的所有顶点都拥有偶数的度数，那么这样的图称为欧拉图。**

实验 9 查找

1 实验目的

⑴ 掌握顺序查找、二分查找、分块查找哈希表查找的算法。

⑵ 掌握二叉排序树的建立和查找算法。

⑶ 能运用线性表的查找方法解决实际问题。

2 基础练习

【注意】本实验中，所有表元素均具有唯一关键字值。

【练习一】填空：在一个无序表 A 中采用顺序查找算法查找值为 x 的元素。

```
# include <stdio.h>
int search (int A[], int x, int n)
{
    int i=0;
    while (i<n && A[i]!=x)
        i++;
    if (i>=n)
        return_____;
    else
        return _____ ;
}
void main()
{
    int a[]={2, 5, 56, 10, 12, 15, 8, 19, 25, 32}, n, d, i;
    printf（”A 数值\n 下标”）;
    for (i=0；i<10；i++)
        printf（“%3d”,i)；
    printf（“\n 值 “）;
    for (i=0；i<10；i++)
        printf（“%3d”,a[i])；
    printf（“\n 输入值：”）;
    scanf（“%d”,&d)；
    n=search(a, d, 10)；
    if (n>＝0)
        printf（“A[%d]=\n”,d)；
    else
        printf（“%d 未找到\n”,d)；
}
```

【练习二】一个长度为 n（n≤1024）的有序表，其元素分别包含一个整型键值。编写程序：

查找键值为 x 的元素。

【练习三】设有关键字集合：{22，12，13，8，9，20，33，42，44，38，24，48，60，58，74，49，86，53}，为其建立索引表，实现分块查找算法。

【提示】分块查找又称为索引顺序查找。分块查找过程分两步进行：首先用某种查找方法查找索引表（若索引表有序，则采用二分查找检索索引表），以确定待查找数据属于哪一块。然后在块内顺序查找要找的数据。本例可分 3 块建立查找表及其索引表，分块结果是块内无序，块间有序。索引表结点结构中包括：块的起始地址与块内的最大关键字。

3 进阶练习

【练习一】一个长度为 n（n≤1024）的有序表，其元素分别包含一个整型键值。编写程序：向有序表中插入一个元素 x，并保持表的有序性。

【提示】利用二分查找方法寻找 x 的插入位置。

【练习二】编程，有 n（n≤100）个整数，构造其对应的二叉排序树。

【提示】二叉排序树的生成，可从空的二叉树开始，每输入一个结点数据，就建立一个新结点插入到当前已生成的二叉排序树中，所以它的主要操作是二叉排序树的插入运算。在二叉排序树中插入新结点，只要保证插入后仍符合二叉排序树的定义即可。

插入新结点的过程：

若二叉排序树为空，则待插入结点 s（结点由指针 s 所指向）作为根结点插入到空树中；当二义排序树非空，将待插结点的关键字 s－>key 与树根的关键字 t->key 比较，若 s->key＝t->key，则说明树中已有此结点，无需插入；右 s->key<t->kly，则将待插结点 s 插入到根的左子树中，否则将 s 插入到根的右子树中。而子树中的插入过程又和在树中的插入过程相同，如此进行下去，直到把结点 s 作为一个新的树叶插入到二叉排序树中，或者直到发现树中已有结点 s 为止。

4 扩展练习

【练习一】哈希表查找。已知一组关键字为(26，36，41，38，44，15，68，12，06，51)，

【问题】编写程序，构造这组关键字的哈希表。要求：用除留余数法构造哈希函数，用线性探测法寻找开放地址，以解决冲突，。

【提示】为减少冲突，通常令装填因子 α<1。这里关键字个数 n=10，不妨取 m=13，此时 α≈0.77，散列表为 T[0..12]，由此得到哈希函数为：h(key)=key％13。

【练习二】现在，你要从滑铁卢去另一个城市，问题是你和那里的人语言不相通。幸好你有一本字典，必要时可以通过查字典进行交流。如下是一些案例。其中，每一行的第一个字符串是英语单词，空格后面的字符串是对应的外语方言。

dog ogday

cat atcay

pig igpay

froot ootfray

loops oopslay

【问题】编写程序，对你所遇到的外语单词，通过查字典将其翻译成英文单词，如果某个单

词在你的字典中找不到，则将其翻译为”eh”。如：

atcay

ittenkay

oopslay

翻译为：

cat

eh

loops

【提示】可将字典建立为磁盘文件。程序首先将磁盘文件中的字典读入内存，在内存中建立字典表，用二叉排序树存储。二叉排序树的每个结点中至少应包含“方言单词”和“英文单词”。注意思考二叉排序树应该以什么为序来建立？当建立好二叉排序字典树之后，翻译就是基于对二叉排序树所进行的搜索。

实验 10 排序

1 实验目的

⑴ 掌握各种不同排序方法的基本思想、排序过程、实现的算法和优缺点。

⑵ 了解各种排序方法依据的原则，以便根据不同的情况选择合适的排序方法。

2 基础练习

【练习一】填空，完成快速排序算法。

```
#defind m 100
typedef struct
{
    int key;
    int no;
}rectype;
typedef  rectype  seqlist[m];
seqlist  r;
int partition(seqlist r , int i, int j)
{
    rectype pivot = r[i];
    while(i<j)
    {
        while(i<j&&r[j].key>=pivot.key)
            _____;
        if(i<j)
            r[_____] = r[j];
        while(i<j&&r[i].key<=pivot.key)
            _____;
        if(i<j)
            r[_____] = r[i];
     }
     _____;
     return i;
}
void quicksort(seqlist r, int low , int high)
{
     int pivotpos;
     if (low<high)
     {
          pivotpos= partition(r,low,high);
          quicksort(r,low,pivotpos-1);
          quicksort(r,piotpos+1,high);
```

```
    }
}
void  main()
{
    int i,n;
    scanf（“%d”,&n);
    for(i=1;i<=n;i++)
        scanf（“%d%d”,&r[i].no,&r[i],key);
    quicksort(r, _____,n);
    for(i=1;i<=n;i++)
        printf（“%d:%d”,r[i].no,r[i].key);
    printf（“\n”);
}
```

【练习二】顺序表中包含 n（n≤100）个 1000 以内的随机整数，假设其中没有重复键值的元素。试对该顺序表按关键字递增的顺序排序。

要求分别使用：直接插入排序、选择排序、冒泡排序、快速排序算法实现。

【练习三】在有 n（n≤80）个学生的成绩表里，每条信息由学号、姓名、数学成绩、英语成绩、计算机成绩与平均成绩组成。要求：

⑴ 按分数高低次序，输出每个学生的名次，分数相同的为同一名次。

⑵ 按名次输出每个学生的姓名与分数。

3 进阶练习

【练习一】顺序表中包含 n（n≤100）个 1000 以内的随机整数，设其中元素没有重复键值。

【问题】试对该顺序表按关键字递增的顺序排序。要求分别用：shell 排序、二路归并排序算法实现。

【练习二】有 n（n<=100）个 1000 以内的随机整数，假设其中没有重复键值的元素。

【问题】试对该顺序表按关键字递增的顺序排序。要求使用堆排序算法。

4 扩展练习

【练习一】哪些饮料最热销？

校园里的小超市或许想知道哪些饮料在哪部分学生群体（男生、女生）中最畅销。可以发起一个调查。根据调查结果，就能大致了解在学生群体中最热销的饮料。

【问题】做一个调查问卷。根据调查结果：

⑴ 找出最受欢迎前三位的饮料。

⑵ 根据购买者的群体（男生、女生）不同，分析不同学生群体的喜好是否有显著差异？

⑶ 根据饮料的类别（碳酸饮料，非碳酸饮料），分析不同学生群体的喜好是否有显著差异？

【练习二】现在的上机考试虽然有实时的评判系统，但上面的排名只是根据完成的题目数排序，没有考虑每题的分值，所以并不是最后的排名。现在给定录取分数线，请你写程序找出最后通过分数线的考生，并将他们的成绩按降序打印。

已知考生人数 N (0 < N < 1000)，考题数目 M (0 < M < = 10)、分数线（正整数）G 为 25 分。每一名考生的准考证号是长度不超过 20 的字符串。如下是一组测试案例：

考生人数：4
题目总数：5
录取分数线：25
每题分值：10,10,12,13,15
学生学号及完成题目情况如表 10-13 所示。

表 10-13 测试案例表

准考证号	解题总数	完成的题目号				
CS004	3	5	1	3		
CS003	5	2	4	1	3	5
CS002	2	1	2			
CS001	3	2	3	5		

【问题】编写程序：按分数从高到低输出上线考生的考号与分数。若有多名考生分数相同，则按他们考号的升序输出。

实验 11 递归

1 实验目的

⑴ 了解递归过程运行的机制。

⑵ 了解递归模型的分析、建立方法，熟练掌握递归程序的编码和调试。

2 基础练习

【练习一】编写一个函数，使用递归的方法计算某一整数 n 的阶乘。

【练习二】斐波纳契数列（Fibonacci Sequence）又称黄金分割数列，是意大利数学家列昂纳多·斐波那契发明的。斐波那契数列指的是这样一个数列：1、1、2、3、5、8、13、21、…。这个数列从第三项开始，每一项都等于前两项之和。在数学上，斐波纳契数列以如下递归的方法定义：

F(0)=0，

F(1)=1，

F(n)=F(n-1)+F(n-2)（n>=2，n∈N*）；

斐波纳契数列在现代物理、准晶体结构、化学等领域都有直接的应用

【问题】编写程序，求斐波纳契数列的第 n 项。

【练习三】编程，使用递归的方法检查某一整数是否是素数。

3 进阶练习

【练习一】编程，找出 100 以内（十进制）的所有快乐数。

【提示】快乐数（happy number）有以下的特性：在给定的进位制下，该数字所有数位的平方和，得到一个新数。对新得到的数再次求所有数位的平方和，如此重复进行，最终结果必为 1。

以十进制为例：

⑴ $28 \rightarrow 2^2+8^2=68 \rightarrow 6^2+8^2=100 \rightarrow 1^2+0^2+0^2=1$

⑵ $32 \rightarrow 3^2+2^2=13 \rightarrow 1^2+3^2=10 \rightarrow 1^2+0^2=1$

⑶ $37 \rightarrow 3^2+7^2=58 \rightarrow 5^2+8^2=89 \rightarrow 8^2+9^2=145$

$\rightarrow 1^2+4^2+5^2=42 \rightarrow 4^2+2^2=20 \rightarrow 2^2+0^2=4$

$\rightarrow 4^2=16 \rightarrow 1^2+6^2=37$……

因此 28 和 32 是快乐数，而在 37 的计算过程中，37 重复出现，继续计算的结果只会是上述数字的循环，不会出现 1，因此 37 不是快乐数。

不是快乐数的数称为不快乐数（unhappy number），所有不快乐数的数位平方和计算，最后都会进入 4→16→37→58→89→145→42→20→4 的循环中。

在十进制下，100 以内的快乐数有：

1, 7, 10, 13, 19, 23, 28, 31, 32, 44, 49, 68, 70, 79, 82, 86, 91, 94, 97, 100。

【练习二】编写递归函数，生成帕斯卡(Pascal)三角形中的数。反复调用该递归函数，可得到一个 n 行的 Pascal 三角形。

【提示】Pascal 三角形在中国也称为杨辉三角形，其样式为：

1

1　1

1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
……

4 扩展练习

【练习】Sierpinski 三角形是一种分形图形，它是递归构造的。最常见的构造方法如图 10-10 所示。

图 10-10 Sierpinski 三角形

它是把一个三角形分成四等份，挖掉中间那一份，然后继续对另外三个三角形进行这样的操作，并且无限地递归下去。每一次迭代后整个图形的面积都会减小到原来的 3/4，因此最终得到的图形面积显然为 0。

【问题】编程，生成一个 Sierpinski 分形模型。

【提示】Sierpinski 三角形模型能够通过 Pascal 三角形生成。在 Pascal 三角形的每一个偶数位置放上 1，每一个奇数位置放上 0，就生成了 Sierpinski 三角形。如图 10-11 所示。

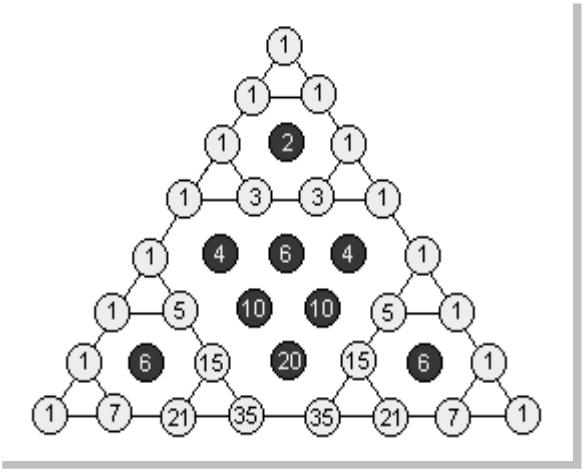


图 10-11 Sierpinski 三角形