**工程实践I 第二部分**

**模块化程序设计训练**

**内容**

为了让大家更好的掌握模块化程序设计方法，及在此基础上能更好得理解同一问题分别采用数组与链表处理时各自的特点，这里给出两个简单应用：“求两个集合的并运算”与“两个有序表合并后仍然有序”，要求编程实现。

这里解决问题的算法框架及调用的函数及功能均做了严格限定，要求分别用数组法与链表法实现，并编写主函数完成功能验证。

**目标**

认真完成工程实践I的第2部分内容，期望实现以下几方面的目标：

1. 希望借由完成严格限定功能及参数的函数这一方式，学习并训练合理的函数功能划分及设计与实现。
2. 这些函数将工程实践I的第二部分中大量使用，本部分即为后续内容做基本准备，以规范完成大作品时的函数功能划分及设计与实现。
3. 此部分内容描述与C语言教材风格差异较大，实际上该内容完整取材于《数据结构》教材。希望反复阅读该内容并适应，为后续数据结构课程学习能快速进入状态做准备。

**检查方式**

以示例讲述实现思想，及采用一一问答、画示意图讲述程序运行过程中变量变化过程方式进行，以确定是否切实理解程序运行过程。

**成绩计算**

此部分得分将占《工程实践1》平时成绩的40%（答辩检查得分20%+程序模块化报告得分20%）。

因为学生量大，检查过程耗时，且需要后续安排足够的时间完成综合性作品，所以实际操作过程中希望同学间有互助精神，并能通过多交流提高效率。基于此，课程组决定具体操作过程分两个批次，其中：

第一批次检查在题目讲解一周后进行，并依据此次检查成绩分小组，第一批通过者得分很高，并委任小组长，组织及指导小组其他成员继续完成，以分担老师一部分工作的基础上锻炼自已的综合能力；（注意第一阶段测试得分43分及以上同学必须参加第一批检查，其他同学根据完成情况也可参加第一批检查）

第二批次在题目公布后第二周进行，方式为小组长先对每个成员打分，然后老师随机抽小组内成员检查，老师检查成绩与小组长检查成绩取分差，其他组员也按该分差进行浮动；组长在自身原有成绩上再加指导分（小组分/10）。

**题目及内容要求（该部分内容教材46-52页）**

**题目一 求两个集合的合并运算**

编程实现两个集合的并运算，即A=A∪B，要求分别采用数组与链表实现，并分析两种方法各自的优缺点。

解决该问题，可以从集合B中依次取得每个数据元素，将其中在集合A中依次查找，若不存在则在A集合中插入该元素。不论采用数组法还是链表法，严格限定实现该算法的框架如下：

void Union(List **&**La,List Lb)

{ // 将所有在表Lb（代表B集合）中但不在La（代表A集合）中的数据元素插入到La中

La\_len=ListLength(La); // 求表La的长度

Lb\_len=ListLength(Lb);

for(i=1;i<=Lb\_len;i++)

{

GetElem(Lb,i,e); // 取表Lb中第i个数据元素赋给变量e

if(!LocateElem(La,e,equal)) // 表La中不存在和e相同的元素,则将e插入La中

ListInsert(La,++La\_len,e);

}

}

**要求：**

1. 结合自己对算法的分析及理解补充表类型“List”的定义，函数局部变量“i”，“e”等的定义，形参“&”可以改为指针外，不允许对程序做其它任何修改；这里的ListLength()、GetElem()、LocateElem()、ListInsert()函数需要自己实现，且针对数组法与链表法需分别编写；自己设计完成主函数。希望这种限定，让大家能很好的理解模块化程序设计。
2. 对表的主要操作函数有如下限定。其中**这里形参中的&不是地址符**（在C语言中只有实参能使用地址符，形参并不能使用地址符）**，是指当该函数让此变量发生变化时，对应的实参量发生同样的改变**，含义更接近C++的引用参数。采用C语言实现时，需改为指针。
3. 求表的长度函数 ListLength(L)

初始条件：表L已存在。

操作结果：返回L中数据元素个数。

1. 取表中的一个元素函数GetElem( L,i,**&**e)

初始条件：表L已存在，1≤i≤ListLength(L)。

操作结果：用e返回L中第i个数据元素的值。

1. 判断表L中符合条件的元素位序函数 LocateElem( L,e,compare( ))

初始条件：表L已存在，compare()是数据元素判定函数(满足为1,否则为0)。

操作结果：返回L中第1个与e满足关系compare()的数据元素的位序。若这样的数据元素不存在，则返回值为0。

1. 在表中插入一个元素函数 ListInsert(**&**L,i,e)

初始条件：表L已存在，1≤i≤ListLength(L)+1。

操作结果：在L中第i个位置之前插入新的数据元素e，L的长度加1。

1. 判断表是否空函数ListEmpty(L)

初始条件：表L已存在。

操作结果：若L为空表，则返回TRUE，否则返回FALSE。

实现时采用的预定义的常量与类型含义如下：

#define OK 1

#define ERROR 0

#define TRUE 1

#define FALSE 0

//Status 为函数的类型，其值是函数结果状态代码，如OK等

typedef int **Status**;

// ElemType为数据元素类型，根据实际情况而定，这里假设为int

typedef int **ElemType**;

1. 对表的存储分别采用数组和链表两种方法实现，每种方法存储上的结构进行了如下限定(为了区分表所采用的存储类型，将采用数组法时的数据表类型命名SqList，将采用链表法的数据表类型名命名为LinkList)：
2. 采用数组实现时的类型定义限定

该应用中需反复多次计算表的长度，针对此背景所设计的存储结构中除了存放数据的一维数组data外，增加一个整型量length用于随时记录表中元素的个数，将两项组成一个结构体。具体定义如下：

#define MAXSIZE 50 /\* 存储空间初始分配量 \*/

typedef struct

{

ElemType data[MAXSIZE]; /\* 数组，存储数据元素 \*/

int length; /\* 表当前有效长度 \*/

}SqList;

如果定义：SqList L，则L的存储示意图1所示。

图1 SqList类型的变量存储结构示意图

L

length

data

... ...

需要表中元素个数信息时，可以直接从L.length域中读出值获取。

如在这种结构下编写前面“要求（二）”所指定的“取表中的一个元素函数GetElem( L,i, &e)”时，参考代码为：

Status GetElem(SqList L,int i,ElemType **\*e**)

{/\* 初始条件：数组表示的表L已存在，1≤i≤ListLength(L) \*/

/\* 操作结果：用e返回L中第i个数据元素的值,

注意i是指位置，第1个位置的数组是从0开始 \*/

if(L.length==0 || i<1 || i>L.length)

return ERROR;

\*e=L.data[i-1];

return OK;

}

特别注意函数实现这里参数的表现方式。因为GetElem( L,i, **&e**)函数的功能是：从表L中取出其中的第i个元素用参数e返回，在C语言想要通过参数返回值需采用指针实现，所以做了如上修改。其它函数的实现参照此函数的实现方式。

另外，“判断表是否空函数ListEmpty(L)”实现参考代码为：

Status ListEmpty(SqList L)

{ /\* 初始条件：数组表示的表L已存在。\*/

/\* 操作结果：若L为空表，则返回TRUE，否则返回FALSE。 \*/

if(L.length==0)

return TRUE;

else

return FALSE;

}

其它函数参照如上代码实现。

“Union函数”可以按如下方式修改实现（也可以不按该方式，允许有自己的自认为更合理的细节实现方案（主要指时间空间更优），只要框架符合题目要求即可）。

void Union(SqList \*La,SqList Lb)

{ /\* 将所有在表Lb中但不在La中的数据元素插入到La中 \*/

ElemType e;

int La\_len,Lb\_len;

int i;

La\_len=ListLength(\*La); /\* 求表La的长度 \*/

Lb\_len=ListLength(Lb);

for(i=1;i<=Lb\_len;i++)

{

GetElem(Lb,i,&e); /\* 取Lb中第i个数据元素赋给e \*/

if(!LocateElem(\*La,e,equal)) /\* La中不存在和e相同的元素,则插入之 \*/

ListInsert(La,++La\_len,e);

}

}

（2）采用链表实现时的类型定义限定

如果定义LinkList L，则L仅是一个指针，并没有实际的数据空间，需要编写函数创建链表并读入数据。假设已经建立的链表如图2所示：

struct LNode /\* 结点定义 \*/

{

ElemType data;

struct LNode \*next;

};

typedef struct LNode \*LinkList; /\* 表的头指针类型 \*/

L

^…

……

图2 链表存储方式示意图

在这种结构下编写“取表中的一个元素函数GetElem( L,i, &e)”,及“判断表是否为空的函数ListEmpty(LinkList L)”可以为：

Status GetElem(LinkList L,int i,ElemType **\*e**)

{ /\* L为单链表的头指针。当第i个元素存在时,

其值赋给e并返回OK,否则返回ERROR \*/

int j=1; /\* j为计数器 \*/

LinkList p=L; /\* p指向第一个结点 \*/

while(p&&j<i) /\* 顺指针向后查找, 直到p指向第i个元素

或p为空 \*/

{

p=p->next;

j++;

}

if(!p||j>i) /\* 第i个元素不存在 \*/

return ERROR;

\*e=p->data; /\* 取第i个元素 \*/

return OK;

}

Status ListEmpty(LinkList L)

{ /\* 初始条件：链式存储的表L已存在。\*/

/\*操作结果：若L为空表，则返回TRUE，否则返回FALSE \*/

if(L->next)

return FALSE;

else

return TRUE;

}

可以看出，对于相同的功能，采用数组与链表因为存储结构不同而使得实现时代码（算法）完全不同。因此，对此应用，采用数组法与链表法时所有函数均各自独立编写，不能重用。

1. 分析此题采用数组与链表时哪种方法更好?说明理由。

**题目二 求两个有序表合并算法**

编程实现将两个有序表合并后仍然有序功能，要求分别采用数组法与链表法，并分析两种方法各自的优缺点。

若用表La、Lb分别代表两个已存在的有序表，Lc为算法完成后产生新的有序表。可行的算法之一为：从表La与Lb中各取一个元素进行比较，将小的元素插入到Lc中，并取小元素所在表的下一个元素继续与另一表的元素继续比较操作，直到一个表中元素取尽为止，再将另一表的余下元素直接挂入新表Lc的末尾。限定本题的算法过程如下：

void MergeList(List La,List Lb,List &Lc)

{ /\* 已知表La和Lb中的数据元素按值非递减排列。 \*/

/\* 归并La和Lb得到新的表Lc,Lc的数据元素也按值非递减排列 \*/

i=j=1;k=0;

La\_len=ListLength(La);

Lb\_len=ListLength(Lb);

while(i<=La\_len&&j<=Lb\_len) /\* 表La和表Lb均非空 \*/

{

GetElem(La,i,ai);

GetElem(Lb,j,bj);

if(ai<=bj)

{

ListInsert(Lc,++k,ai);

++i;

}

else

{

ListInsert(Lc,++k,bj);

++j;

}

}

while(i<=La\_len) /\* 表La非空且表Lb空 \*/

{

GetElem(La,i++,ai);

ListInsert(Lc,++k,ai);

}

while(j<=Lb\_len) /\* 表Lb非空且表La空 \*/

{

GetElem(Lb,j++,bj);

ListInsert(Lc,++k,bj);

}

}

**要求：**与题一相同。

题一所编写的对表进行基本操作的各函数这里可以直接使用，不用重新编写。建议将这些函数写入一个文件形成函数库，使用时以程序以头文件的形式加入，方便重用。

通过完成这两个严格限定的应用可以发现，对于同一个问题，采用数组法与链表法两种存储方式实现时，各函数需分别编写，不能重用；但对不同的应用，采用同一存储方式的函数可以重用。这种经验的积累，为后续从数据结构角度解决编程问题的思维学习打下基础。