**本书中的算法实现模板（C语言）**

数据结构的学习既有很强的理论性还有较强的实践性。书中算法描述采用的都是标准Ｃ函数。当需要在计算机上完整地实现算法时，必须设计构造一个完整的可以执行的源程序。

下面给出单文件算法实现模板和多文件算法实现模板。

[程序实例]：将元素非递减的顺序表LA和LB，合并为LC，要求合并后的LC仍为非递减。

**模板一：单文件结构模板**

考虑到初学者对多文件结构不熟悉，首选给出一个单文件结构的算法实现模板。

#include <stdio.h> /\*包含C库头文件\*/

#include <stdlib.h>

#define MAXSIZE 100 /\* 必要的宏定义\*/

#define OK 1

#define ERROR 0

typedef int ElemType; //假设顺序表中的数据元素是整型

typedef struct{

ElemType elem[MAXSIZE];

int last;

}SeqList;

/\*函数声明\*/

void InitList(SeqList \*L);

int InsList(SeqList \*L,int i,ElemType e);

void InputList(SeqList \*L);

void OutputList(SeqList L);

void merge(SeqList \*LA, SeqList \*LB, SeqList \*LC);

int main()

{

SeqList L1,L2,L3; /\*定义三个顺序表\*/

InitList(&L1); /\*分别予以初始化\*/

InitList(&L2);

InitList(&L3);

printf("\nL1: 按非递减顺序输入若干元素，空格分开，数目<=50，以-100结束\n");

InputList(&L1);

printf("\nL2: 按非递减顺序输入若干元素，空格分开，数目<=50，以-100结束\n");

InputList(&L2);

merge(&L1,&L2,&L3);

printf("\n顺序表L1和L2合并后的结果为：\n");

OutputList(L3);

return 0;

}

/\*初始化顺序表\*/

void InitList(SeqList \*L)

{

L->last=-1;

}

/\*在顺序表L中第i个数据元素之前插入一个元素e。 插入前表长n=L->last+1，i的合法取值范围是 1≤i≤L->last+2 \*/

int InsList(SeqList \*L,int i,ElemType e)

{

int k;

if((i<1) || (i>L->last+2)) /\*首先判断插入位置是否合法\*/

{

printf("插入位置i值不合法");

return(ERROR);

}

if(L->last>= MAXSIZE-1)

{

printf("表已满无法插入");

return(ERROR);

}

for(k=L->last;k>=i-1;k--) /\*为插入元素而移动位置\*/

L->elem[k+1]=L->elem[k];

L->elem[i-1]=e; /\*在C语言数组中，第i个元素的下标为i-1\*/

L->last++;

return(OK);

}

/\*为顺序表赋值\*/

void InputList(SeqList \*L)

{

ElemType d;

scanf("%d",&d);

while(d!=-100)

{

InsList(L,L->last+2,d); /\*调用插入操作，将新输入的元素放入顺序表的末尾\*/

scanf("%d",&d);

}

}

/\*输出顺序表中的元素\*/

void OutputList(SeqList L)

{

int i;

for(i=0;i<=L.last;i++)

printf("%d ",L.elem[i]);

}

/\*将元素非递减的顺序表LA和LB，合并为LC。合并后的LC仍为非递减\*/

void merge(SeqList \*LA, SeqList \*LB, SeqList \*LC)

{

int i,j,k;

i=0;j=0;k=0;

while(i<=LA->last&&j<=LB->last)

if(LA->elem[i]<=LB->elem[j])

{

LC->elem[k]= LA->elem[i];

i++;

k++;

}

else

{

LC->elem[k]=LB->elem[j];

j++;

k++;

}

while(i<=LA->last) /\*当表LA有剩余元素时，则将表LA余下的元素赋给表LC\*/

{

LC->elem[k]= LA->elem[i];

i++;

k++;

}

while(j<=LB->last) /\*当表LB有剩余元素时，则将表LB余下的元素赋给表LC\*/

{

LC->elem[k]= LB->elem[j];

j++;

k++;

}

LC->last=LA->last+LB->last+1;

}

**模板二：多文件结构模板**

大型程序一般都采用多文件结构，以下是多文件结构的顺序表合并的例子。

模板的基本结构如下：

⑴　包含必要的标准头文件，如标准的输入输出头文件"stdio.h"；同时给出必要的符号常量宏定义；

⑵　将某一数据结构所对应的类型定义存放在一个头文件当中，将某一数据结构所对应的基本操作算法存放在一个分类的.c文件当中.如：可以将单链表的有关类型定义存放在seqlist.h中,将单链表的基本操作算法存放在seqlist.c中，之后通过文件包含#include "seqlist.h" 和#include "seqlist.c",以实现对有关数据类型的引用及有关操作函数的调用；

⑶　编写基于某种数据结构的具体问题的算法；

⑷　编写主函数，其中进行合理的函数调用，形成一个可执行程序。

**/\*main.cpp\*/**

#include "stdio.h" /\*包含标准输入输出文件\*/

#include "seqlist.cpp" /\*包含顺序表文件\*/

#define OK 1 /\* 必要的宏定义\*/

#define ERROR 0

/\*函数声明\*/

void merge(SeqList \*LA, SeqList \*LB, SeqList \*LC);

int main()

{

SeqList L1,L2,L3; /\*定义三个顺序表\*/

InitList(&L1); /\*分别予以初始化\*/

InitList(&L2);

InitList(&L3);

printf("\nL1: 按非递减顺序输入若干元素，空格分开，数目<=50，以-100结束\n");

InputList(&L1);

printf("\nL2: 按非递减顺序输入若干元素，空格分开，数目<=50，以-100结束\n");

InputList(&L2);

merge(&L1,&L2,&L3);

printf("\n顺序表L1和L2合并后的结果为：\n");

OutputList(L3);

return 0;

}

/\*将元素非递减的顺序表LA和LB，合并为LC。合并后的LC仍为非递减\*/

void merge(SeqList \*LA, SeqList \*LB, SeqList \*LC)

{

int i,j,k;

i=0;j=0;k=0;

while(i<=LA->last&&j<=LB->last)

if(LA->elem[i]<=LB->elem[j])

{

LC->elem[k]= LA->elem[i];

i++;

k++;

}

else

{

LC->elem[k]=LB->elem[j];

j++;

k++;

}

while(i<=LA->last) /\*当表LA有剩余元素时，则将表LA余下的元素赋给表LC\*/

{

LC->elem[k]= LA->elem[i];

i++;

k++;

}

while(j<=LB->last) /\*当表LB有剩余元素时，则将表LB余下的元素赋给表LC\*/

{

LC->elem[k]= LB->elem[j];

j++;

k++;

}

LC->last=LA->last+LB->last+1;

}

**/\*seqlist.h\*/**

#define MAXSIZE 100

typedef int ElemType; //假设顺序表中的数据元素是整型

typedef struct{

ElemType elem[MAXSIZE];

int last;

}SeqList;

**/\*seqlist.cpp\*/**

#include "stdio.h"

#include "seqlist.h"

#define OK 1 /\* 必要的宏定义\*/

#define ERROR 0

/\*初始化顺序表\*/

void InitList(SeqList \*L)

{

L->last=-1;

}

/\*在顺序表L中第i个数据元素之前插入一个元素e。 插入前表长n=L->last+1，i的合法取值范围是 1≤i≤L->last+2 \*/

int InsList(SeqList \*L,int i,ElemType e)

{

int k;

if((i<1) || (i>L->last+2)) /\*首先判断插入位置是否合法\*/

{

printf("插入位置i值不合法");

return(ERROR);

}

if(L->last>= MAXSIZE-1)

{

printf("表已满无法插入");

return(ERROR);

}

for(k=L->last;k>=i-1;k--) /\*为插入元素而移动位置\*/

L->elem[k+1]=L->elem[k];

L->elem[i-1]=e; /\*在C语言数组中，第i个元素的下标为i-1\*/

L->last++;

return(OK);

}

/\*为顺序表赋值\*/

void InputList(SeqList \*L)

{

ElemType d;

scanf("%d",&d);

while(d!=-100)

{

InsList(L,L->last+2,d); /\*调用插入操作，将新输入的元素放入顺序表的末尾\*/

scanf("%d",&d);

}

}

/\*输出顺序表中的元素\*/

void OutputList(SeqList L)

{

int i;

for(i=0;i<=L.last;i++)

printf("%d ",L.elem[i]);

}