# 静态顺序存储线性表的基本实现

**任务描述：**

        根据给出的存储结构，完成现在下列函数功能，并按照主函数的代码，输出规定的效果。

**预定义常量和类型：**

#defineLIST\_INITSIZE100

#defineElemTypeint

#defineStatusint

#defineOK1

#defineERROR0

**存储结构：**

typedefstruct

{

ElemTypeelem[LIST\_INITSIZE];

intlength;

}SqList;

**函数操作：**

输出打印：voidListPrint(SqListL);

删除：void DeleteElem(SqList \*L);

**注意：**

对于输出打印是指将表中**数据元素打印**出来；

删除操作是**删除比第三个元素大的且比第六个元素小的数据元素**,并保证元素的相对位置不变。

**主函数如下：**

int main(void)

{

SqList L;

int i = 0;

ElemType e;

ElemType data[9] = {10,263,-32,-3,-25,88,77,0,-9};

InitList\_Sq(&L);

for(i = 1;i <= 9;i++)

ListInsert(&L,i,data[i-1]);

printf("插入完成后L = : ");

ListPrint(L);

DeleteElem(&L);

printf("删除比第三个元素大的且比第六个元素小的数据元素： ");

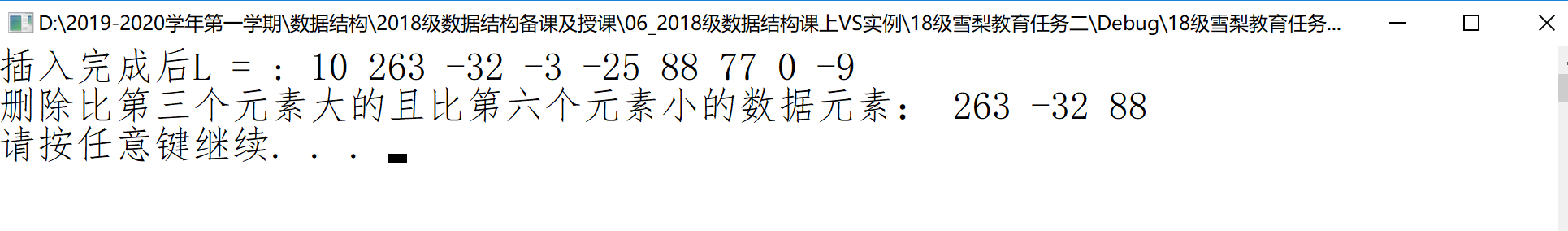
ListPrint(L);

system("pause");

return 0;

}

运行效果：



**编程要求：**

1. 按照题目要求，用C语言编写完整程序，并按输出效果输出；
2. 必须提交ListPrint和DeleteElem函数代码；
3. 严格按照题目所给的存储结构实现；
4. **如果该程序输出结果不对至少扣一半的分**。

**提交格式：**

ListPrint函数如下：

voidListPrint(SqListL){

}

DeleteElem函数如下：

void DeleteElem(SqList \*L)

{

}

# 动态顺序存储线性表的基本实现

**任务描述：**

根据给出的存储结构，完成现在下列函数功能，并按照主函数的代码，输出规定的效果。

**预定义常量和类型：**

#defineLIST\_INIT\_SIZE100

#defineLISTINCREMENT10

#defineStatusint

#defineOVERFLOW-1

#defineOK1

#defineERROR0

#defineElemTypeint

**存储结构：**

typedefstruct

{

ElemType\*elem;

intlength;

intlistsize;

}SqList;

**函数操作：**

**打印：**voidListPrint\_Sq(SqListL)

分类：void PartList(SqList \*La)

**注意：**

对于输出打印(ListPrint\_Sq)是指将表中**数据元素打印**出来；分类(PartList)是在线性表L中**将正数和负数分开，正数在前，负数在后，0在中间，正数的相对位置保持不变，负数的相对位置也保持不变。**

**主函数如下：**

int main(void)

{

SqList L;

int i=0;

ElemType e;

ElemType data[9] = {10,263,-32,-3,-25,88,77,0,-9};

InitList\_Sq(&L);

for(i = 0;i < 9; i++)

ListInsert\_Sq(&L,i+1,data[i]);

printf("插入完成后 L = : ");

ListPrint\_Sq(L);

PartList(&L);

printf("元素分类后的线性表：");

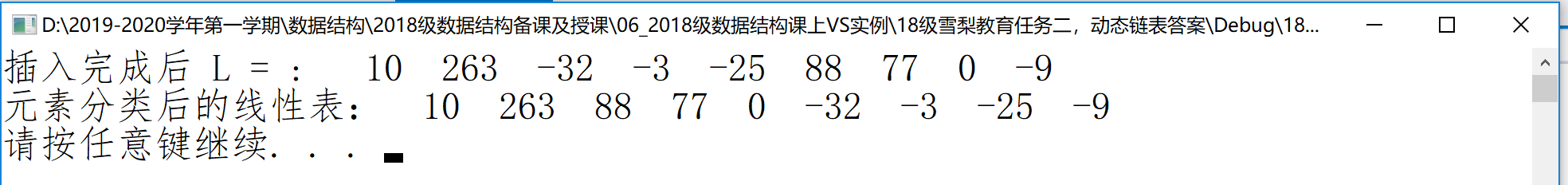
ListPrint\_Sq(L);

system("pause");

return 0;

}

运行效果：



**编程要求：**

1. 按照题目要求，用C语言编写实现每个函数，并按输出效果输出；
2. 必须提交可执行代码；
3. 函数之间可以相互调用；
4. 严格按照题目所给的存储结构实现；
5. **如果该程序输出结果不对至少扣一半的分**。

**提交格式：**

ListPrint\_Sq函数如下：

void ListPrint\_Sq(SqList L){

}

PartList函数如下：

void PartList(SqList \*La){

}

# 2-1 答案

#include <stdio.h>

#include <stdlib.h>

#define LIST\_INITSIZE 100

#define ElemType int

#define Status int

#define OK 1

#define ERROR 0

typedef struct

{

ElemType elem[LIST\_INITSIZE];

int length;

}SqList;

Status InitList\_Sq(SqList \*L)

{

L->length = 0;

return OK;

}

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

{

int k;

if(i < 1 || i > L->length+1)

{

printf("i值不合法！\n");

return ERROR;

}

if(L->length >= LIST\_INITSIZE)

{

printf("元素已满，不能插入！\n");

return ERROR;

}

for(k = L->length-1; k >= i-1; --k)

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

L->elem[i-1] = e;

++L->length;

return OK;

}

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

{

ElemType \*p, \*q;

int k = 0;

if(i < 1 || i > L->length)

{

printf("删除位置有误!\n");

return ERROR;

}

\*e = L->elem[i-1];

for(k = i-1; k < L->length-1; ++k)

{

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

}

--L->length;

return OK;

}

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

{

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

{

printf("获取元素位置有误!\n");

return ERROR;

}

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

return OK;

}

void DeleteElem(SqList \*L)

{

ElemType third, six, e;

int i;

GetElem(\*L,3,&third);

GetElem(\*L,6,&six);

for(i = 1;i <= L->length; i++)

{

GetElem(\*L, i, &e);

if(e > third && e < six)

{

ListDelete(L,i,&e);

i--;

}

}

}

void ListPrint(SqList L)

{

int i;

ElemType e;

for(i = 1; i <= L.length; ++i)

{

if(GetElem(L, i ,&e))

printf("%d ",e);

}

printf("\n");

}

int main(void)

{

SqList L;

int i = 0;

ElemType e;

ElemType data[9] = {10,263,-32,-3,-25,88,77,0,-9};

InitList\_Sq(&L);

for(i = 1;i <= 9;i++)

ListInsert(&L,i,data[i-1]);

printf("插入完成后L = : ");

ListPrint(L);

DeleteElem(&L);

printf("删除比第三个元素大的且比第六个元素小的数据元素： ");

ListPrint(L);

system("pause");

return 0;

}

# 2-2 答案

#include <stdio.h>

#include <stdlib.h>

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define Status int

#define OVERFLOW -1

#define OK 1

#define ERROR 0

#define ElemType int

typedef struct

{

ElemType \* elem;

int length;

int listsize;

}SqList;

int InitList\_Sq(SqList \*L)

{

L->elem = (int \*)malloc((LIST\_INIT\_SIZE)\*sizeof(int));

if(! L->elem) return 0;

L->length = 0;

L->listsize = LIST\_INIT\_SIZE;

return 1;

}

int ListInsert\_Sq(SqList \*L,int i,int e)

{

int \*q,\*p,\*newbase;

if(i < 1 || i > L->length +1 ) return 0;

if(L->length >= L->listsize ){

newbase = (int \*)realloc(L->elem

,(L->listsize + LISTINCREMENT)\*sizeof(int));

if(! newbase) return 0;

L->elem = newbase;

L->listsize += LISTINCREMENT;

}

q = &(L->elem[i - 1]);

for(p = &(L->elem[L->length-1]);p >= q; --p)

\*(p+1) = \*p;

\*q = e;

L->length++;

return 1;

}

int ListDelete\_Sq(SqList \*L,int i,int \*e)

{

int \*p,\*q;

if ((i < 1) || (i > L->length))

return 0;

p = &(L->elem[i-1]);

\*e = \*p;

q = L->elem + L->length-1;

for (++p;p<=q;++p)

\*(p-1) = \*p;

L->length--;

return 1;

}

void ListPrint\_Sq(SqList L)

{

int i = 0;

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

{

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

}

printf(" \n");

}

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

{

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

{

printf("获取元素位置有误!\n");

return ERROR;

}

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

return OK;

}

void PartList(SqList \*La)

{

int i, j;

ElemType e;

for(i = 1, j = 1; j <= La->length; ++i, ++j)

{

GetElem(\*La, i, &e);

if(e < 0)

{

ListDelete\_Sq(La, i, &e);

ListInsert\_Sq(La, La->length + 1,e);

--i;

}

}

}

int main(void)

{

SqList L;

int i=0;

ElemType e;

ElemType data[9] = {10,263,-32,-3,-25,88,77,0,-9};

InitList\_Sq(&L);

for(i = 0;i < 9; i++)

ListInsert\_Sq(&L,i+1,data[i]);

printf("插入完成后 L = : ");

ListPrint\_Sq(L);

PartList(&L);

printf("元素分类后的线性表：");

ListPrint\_Sq(L);

system("pause");

return 0;

}

参考答案二：

# 2-1 答案

# 2-2 答案

#include <stdio.h>

#include <stdlib.h>

#define LIST\_INIT\_SIZE 100

#define LISTINCREMENT 10

#define Status int

#define OVERFLOW -1

#define OK 1

#define ERROR 0

#define ElemType int

typedef struct

{

ElemType \* elem;

int length;

int listsize;

}SqList;

int InitList\_Sq(SqList \*L)

{

L->elem = (int \*)malloc((LIST\_INIT\_SIZE)\*sizeof(int));

if(! L->elem) return 0;

L->length = 0;

L->listsize = LIST\_INIT\_SIZE;

return 1;

}

int ListInsert\_Sq(SqList \*L,int i,int e)

{

int \*q,\*p,\*newbase;

if(i < 1 || i > L->length +1 ) return 0;

if(L->length >= L->listsize ){

newbase = (int \*)realloc(L->elem

,(L->listsize + LISTINCREMENT)\*sizeof(int));

if(! newbase) return 0;

L->elem = newbase;

L->listsize += LISTINCREMENT;

}

q = &(L->elem[i - 1]);

for(p = &(L->elem[L->length-1]);p >= q; --p)

\*(p+1) = \*p;

\*q = e;

L->length++;

return 1;

}

int ListDelete\_Sq(SqList \*L,int i,int \*e)

{

int \*p,\*q;

if ((i < 1) || (i > L->length))

return 0;

p = &(L->elem[i-1]);

\*e = \*p;

q = L->elem + L->length-1;

for (++p;p<=q;++p)

\*(p-1) = \*p;

L->length--;

return 1;

}

void ListPrint\_Sq(SqList L)

{

int i = 0;

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

{

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

}

printf(" \n");

}

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

{

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

{

printf("获取元素位置有误!\n");

return ERROR;

}

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

return OK;

}

void PartList(SqList \*La)

{

int i, j, k = 0;

ElemType e, \*p, \*q;

for(i = 1, j = 1; j <= La->length; ++i, ++j)

{

e = La->elem[i-1];

if(e < 0)

{

p = &(La->elem[i-1]);

q = La->elem + La->length-1;

for (++p; p <= q; ++p)

\*(p-1) = \*p;

La->elem[La->length-1] = e;

--i;

}

}

for(i = 1; i <= La->length; ++i)

if(La->elem[i-1] >= 0)

++k;

for(i = 1, j = 1; j <= k; ++i, ++j)

{

e = La->elem[i-1];

if(e == 0)

{

p = &(La->elem[i-1]);

q = La->elem + k-1;

for (++p; p <= q; ++p)

\*(p-1) = \*p;

La->elem[k-1] = e;

--i;

}

}

}

int main(void)

{

SqList L;

int i=0;

ElemType e;

ElemType data[9] = {10,263,-32,-3,-25,88,77,0,-9};

InitList\_Sq(&L);

for(i = 0;i < 9; i++)

ListInsert\_Sq(&L,i+1,data[i]);

printf("插入完成后 L = : ");

ListPrint\_Sq(L);

PartList(&L);

printf("元素分类后的线性表：");

ListPrint\_Sq(L);

system("pause");

return 0;

}