1. 问题描述与分析

1.已知A,B,C为三个元素值递增有序的线性表，现要求对表A作如下运算：删除那些即在表B中出现又在表C中出现的元素。

2.抽象数据类型定义

typedef struct {

ElemType\* elem;

int length;

int listsize;

}Sqlist;

3输入值范围

这里设计为整形

问题解决思路

我的解决思路是，先找到B和C中那些有交集的元素，抽取出来。在和A中元素比较，没有相等的话一一填入一个新的数组。本来我是想用删除函数。但是想想时间效率在重复元素多的情况下有点低，不如干脆在开辟一个数组空间，转移过去更好。

1. 算法及算法分析
2. 结构体。

#include< stdlib.h>

#include <stdio.h>

typedef int Status;

typedef int ElemType;

#define TRUE 1

#define FALSE 0

#define OK 1

#define ERROR 0

#define INFEASIBLE -1

#define OVERFLOW -2

#define LIST\_INIT\_SIZE 100

#define LISTINCREAMENT 10

typedef struct { //定义线性表的基本结构 \*elem 这个是理解的关键 这个为整个线性表基地址 后续可通过 L.elem[x]来访问

ElemType\* elem; //顺序表第x个元素

int length;

int listsize;

}Sqlist;

Status InitList(Sqlist& L) {

L.elem = (ElemType\*)malloc(LIST\_INIT\_SIZE \* sizeof(ElemType));

if (!L.elem) exit(OVERFLOW);

L.length = 0;

L.listsize = LIST\_INIT\_SIZE;

return OK;

}//初始化线性表

Status GetElem(Sqlist& L, int i, ElemType& e) { //将读到的元素返回给 e

if (i<1 || i>L.length) return ERROR;

e = \*(L.elem + i - 1);

return OK;

}//读表元

Status Input\_List(Sqlist& L, int n) {

if (n > L.listsize) return OVERFLOW;

else

{

for (int i = 0; i < n; i++)

{

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

L.length++;

}

return OK;

}

}

Status Output(Sqlist L) {

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

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

}

printf(" \n");

return OK;

}

void Remake(Sqlist &La, Sqlist Lb, Sqlist Lc) {

ElemType temp[LIST\_INIT\_SIZE];

//int time = (Lb.length < Lc.length ? Lb.length: Lc.length);

int i = 0;

//int j = 1;

ElemType\* pc = Lc.elem;

ElemType\* pb = Lb.elem;

ElemType\* pc\_last = Lc.elem + Lc.length - 1;

ElemType\* pb\_last = Lb.elem + Lb.length - 1;

while (pc <= pc\_last && pb <= pb\_last) {

if (\*pb == \*pc) {

temp[i] = \*pb;

pb++;

pc++;

i++;

}

else {

if (\*pb < \*pc)

pb++;

else

pc++;

}

/\*

if (\*pb == \*pc) {

pb++;

pc++;

temp[i] = \*pb;

i++;

}

\*/

}

ElemType\* pa = (ElemType\*)malloc(LIST\_INIT\_SIZE\*sizeof(ElemType));

int ques=0;

int pa\_i = 0;

for (int n = 0; n < La.length; n++) {

for (int m = 0; m < i; m++) {

if (La.elem[n] == temp[m]) {

ques = 1;

}

}

if (ques == 0) {

pa[pa\_i] = La.elem[n];

pa\_i++;

}

ques = 0;

}

free(La.elem);

La.elem = pa;

La.length = La.length - i;

}

int main() {

printf("以输入五个元素的线性表为测试，线性表A\n"); // 1 3 5 7 9

Sqlist A; //12 15 24 35 69

InitList(A);

printf("请输入五个元素\n");

Input\_List(A, 5);

Output(A);

printf("以输入五个元素的线性表为测试，线性表B\n"); //2 3 6 7 9

Sqlist B; //22 24 33 42 45

InitList(B);

printf("请输入五个元素\n");

Input\_List(B, 5);

Output(B);

printf("以输入五个元素的线性表为测试，线性表C\n"); //3 7 8 10 11

Sqlist C; //1 7 24 28 29

InitList(C);

printf("请输入五个元素\n");

Input\_List(C, 5);

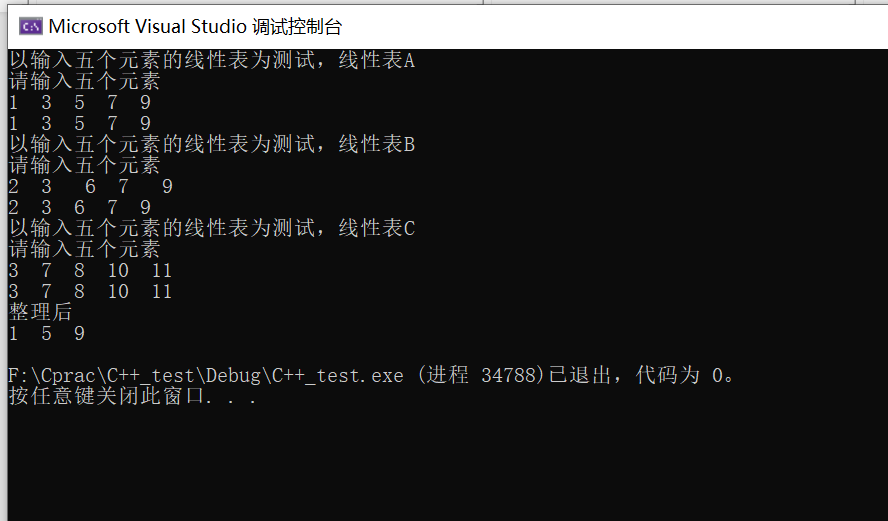
Output(C);

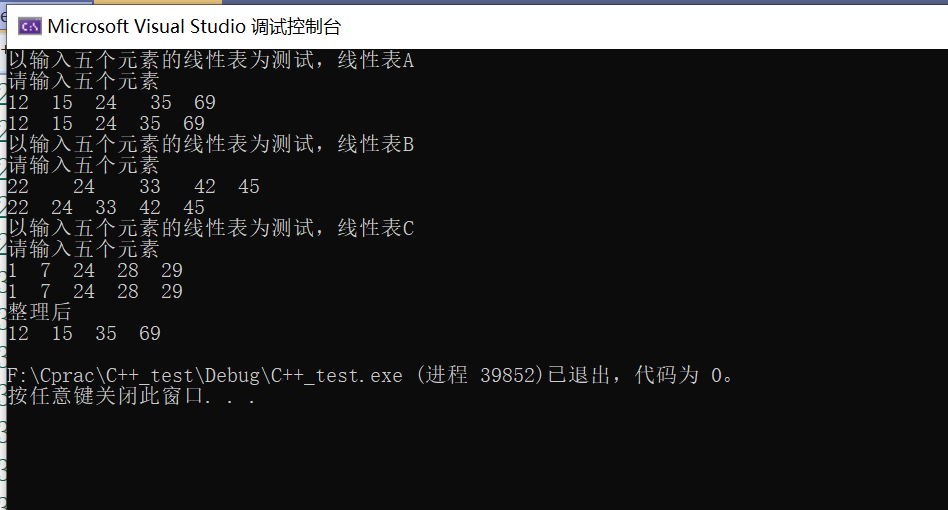
Remake(A, B, C);

printf("整理后\n");

Output(A);

}





算法性能分析：

题目要求的函数 （Remake) 时间复杂度 O(n^2)

空间复杂度 O（n)

不足之处：

感觉逻辑上有点混乱，变量设置比较多，可考虑更清晰简洁一点的算法