有顺序表A和B，其元素均按从小到大的升序排列，编写一个算法，将它们合并成一个顺序表C，要求C的元素也按从小到大的升序排列。

#include "stdio.h"

#include "stdlib.h"

#include "string.h"

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 顺序表的头文件,文件名：数据结构.cpp \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define MAXSIZE 100

typedef int datatype;

typedef struct{

datatype a[MAXSIZE]; //学号

char \*name[MAXSIZE]; // 姓名

datatype score[MAXSIZE]; //数据结构成绩;

int size;

}sequence\_list;

//顺序表的初始化——置空表

void init(sequence\_list \*slt);

//在顺序表后部进行插入操作

void append(sequence\_list \*slt,datatype x,char \*name,int score);

//打印顺序表各节点的值

void display(sequence\_list slt);

//判断顺序表是否为空

int empty(sequence\_list slt);

//查找顺序表中值为x的节点位置

int find(sequence\_list slt,datatype x);

//取得顺序表当中第i个节点的值

datatype get(sequence\_list slt,int i);

//在顺序表的 position 位置插入值为x的节点

void insert(sequence\_list \*slt,datatype x,int position,char \*name,int score);

//删除顺序表的 position 位置的节点

void dele(sequence\_list \*slt,int position);

//顺序表成绩排序

void sort(sequence\_list \*slt);

//交换节点

void exchange(sequence\_list \*slt,int position1,int position2);

//交换int类型的数据

void exchange\_num(datatype \*x,int \*y);

//交换字符串的数据

void exchange\_string(char a[],char b[]);

//顺序表的合成

void sort\_insert(sequence\_list \*slt1,sequence\_list \*slt2,sequence\_list \*slt3);

void sort\_insert2(sequence\_list \*slt1,sequence\_list \*slt2);

int main()

{

sequence\_list Stu1,Stu2,Stu3;

init(&Stu1);

init(&Stu2);

init(&Stu3);

append(&Stu1,1707004711,"小明",56);

append(&Stu1,1707004712,"小华",62);

append(&Stu1,1707004713,"小花",73);

append(&Stu1,1707004714,"钢铁侠",78);

append(&Stu2,1707004715,"王也",43);

append(&Stu2,1707004716,"小浩",56);

append(&Stu2,1707004717,"小逸群",65);

append(&Stu2,1707004718,"小智鹏",72);

append(&Stu2,1707004719,"蜘蛛侠",83);

printf("原始的数据：\n");

printf("数据表1：\n");

display(Stu1);

printf("数据表2：\n");

display(Stu2);

printf("\n-------------------------------------\n");

printf("\n利用slt1跟slt2加到slt3当中的方法:\n\n");

sort\_insert(&Stu1,&Stu2,&Stu3);

display(Stu3);

printf("\n利用slt1插入到slt2的方法:\n\n");

sort\_insert2(&Stu1,&Stu2);

display(Stu2);

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：顺序表的初始化——置空表 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void init(sequence\_list \*slt)

{

slt->size = 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：在顺序表后部进行插入操作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void append(sequence\_list \*slt,datatype x,char \*name,int score)

{

if(slt->size == MAXSIZE)

{

printf("顺序表是满的");

}

slt->a[slt->size] = x;

slt->score[slt->size] = score;

slt->name[slt->size] = name;

slt->size = slt->size + 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：打印顺序表各节点的值 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void display(sequence\_list slt)

{

int i;

if(!slt.size)

printf("\n顺序表是空的！");

else

printf("\t学号\t\t姓名\t数据结构成绩\n");

for(i = 0;i < slt.size;i ++)

printf("%15d\t\t%s\t%7d\n",slt.a[i],slt.name[i],slt.score[i]);

printf("\n");

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：判断顺序表是否为空 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int empty(sequence\_list slt)

{

return(slt.size == 0 ? 1 : 0);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：查找顺序表中值为x的节点位置 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int find(sequence\_list slt,datatype x)

{

int i = 0;

while(i < slt.size && slt.a[i] != x)

i ++;

return(i < slt.size ? i : -1);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：取得顺序表当中第i个节点的学号 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

datatype get(sequence\_list slt,int i)

{

if(i < 0 || i >= slt.size)

{

printf("\n指定位置的节点不存在！");

exit(1);

}

else

return slt.a[i];

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：在顺序表的 position 位置插入值为x的节点 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void insert(sequence\_list \*slt,datatype x,int position,char \*name,int score)

{

int i;

if(slt->size == MAXSIZE)

{

printf("\n顺序表是满的!没法插入!");

exit(1);

}

if(position < 0 || position > slt->size)

{

printf("\n指定的插入位置不存在!");

exit(1);

}

for(i = slt->size;i > position;i --)

{

slt->a[i] = slt->a[i - 1];

slt->name[i] = slt->name[i - 1];

slt->score[i] = slt->score[i - 1];

}

slt->a[position] = x;

slt->name[position] = name;

slt->score[position] = score;

slt->size ++;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：删除顺序表的 position 位置的节点 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void dele(sequence\_list \*slt,int position)

{

int i;

if(slt->size == 0)

{

printf("顺序表是空的!");

exit(1);

}

if(position < 0 || position >= slt->size)

{

printf("\n指定的删除位置不存在");

exit(1);

}

for(i = position;i < slt->size - 1;i ++)

{

slt->a[i] = slt->a[i + 1];

slt->name[i] = slt->name[i + 1];

slt->score[i] = slt->score[i + 1];

}

slt->size --;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：将顺序表slt1跟顺序表slt2顺序连接到stl3当中\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sort\_insert(sequence\_list \*slt1,sequence\_list \*slt2,sequence\_list \*slt3)

{

int i = 0,j = 0; //i用于扫描顺序表slt1,j用于扫描顺序表slt2

if(slt1->size == 0 || slt2->size == 0)

printf("有空的顺序表!\n");

while(i < slt1->size && j < slt2->size)

{

if(slt1->score[i] < slt2->score[j])

{

//在slt3的后面添加slt1的数据

append(slt3,slt1->a[i],slt1->name[i],slt1->score[i]);

//

i ++;

}

else

{

//在slt3的后面添加slt2的数据

append(slt3,slt2->a[j],slt2->name[j],slt2->score[j]);

j ++;

}

}

while(i < slt1->size)//若slt1未扫描完，则将slt1剩余的元素加到slt3中

{

append(slt3,slt1->a[i],slt1->name[i],slt1->score[i]);

i ++;

}

while(j < slt2->size)//若slt2未扫描完，则将slt2剩余的元素加到slt3中

{

append(slt3,slt2->a[j],slt2->name[j],slt2->score[j]);

j ++;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* 函数功能：将顺序表slt1跟顺序表顺序的插到slt2当中\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void sort\_insert2(sequence\_list \*slt1,sequence\_list \*slt2)

{

int i = 0,j = 0;

while(i < slt1->size)

{

//如果顺序表slt1的数据比slt2最大的都大，直接将slt1的数据加到slt2后面

if(slt1->score[i] > slt2->score[slt2->size - 1])

{

append(slt2,slt1->a[i],slt1->name[i],slt1->score[i]);

i ++;

}else if(slt1->score[i] < slt2->score[j]) //如果序表slt1的数据比slt2位置j上的的要小

{

// 将slt1的数据插入到slt2的 j 的位置

insert(slt2,slt1->a[i],j,slt1->name[i],slt1->score[i]);

//两个顺序表都向后扫描

i ++;

j ++;

}else

//向后扫描slt2

j ++;

}

}

结果演示：

