将顺序表(a1,a2,…,an)重新排列以a1为界的两部分：a1前面的值均比a1小，a1后面的值均比a1大（这里假设数据元素的类型具有可比性，不妨设为整型）。

#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);

//将a1作为界限进行划分

void compare(sequence\_list \*slt);

int main()

{

sequence\_list Stu;

init(&Stu);

append(&Stu,1707004711,"小明",73);

append(&Stu,1707004712,"小华",56);

append(&Stu,1707004713,"小花",82);

append(&Stu,1707004714,"小草",78);

append(&Stu,1707004715,"杨旭",74);

append(&Stu,1707004716,"王浩",67);

append(&Stu,1707004717,"丁逸群",95);

append(&Stu,1707004718,"武智鹏",87);

insert(&Stu,1707004710,3,"蜘蛛侠",34);

//首先展示所添加的数据

display(Stu);

//以a1为分界线划分数据

compare(&Stu);

//再次展示顺序表的数据

display(Stu);

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：将顺序表重新排列,a1位置为界

分为两部分，position前面的值均比slt->a[position]的数值小

position后面的值均比slt->a[position]的数值大 \*/

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

void compare(sequence\_list \*slt)

{

if(!slt->size)

printf("\n顺序表是空的！\n");

else

{

//首先保存第一个学生的成绩

int score = slt->score[0];

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

{

if(slt->score[i] < score)

{

//如果遍历学生的成绩大于排头，将该数据插入到第一位

insert(slt,slt->a[i],0,slt->name[i],slt->score[i]);

//删除刚刚移动的那个原数据

dele(slt,i + 1);

}

}

}

}结果演示：

