**实验报告（6.1）**

**姓 名： 鲁日中天 学 号： 2016211339**

**班 级： 2016211308 日 期：2017/12/30**

**分工情况：个人完成**

上机作业

# 实验6.1

目标：

查找表是学生记录的集合，记录由如下数据项组成：  
 （学号，姓名，数据结构课程成绩）  
存储结构为双向链表。  
(1)用快速排序的递归算法对查找表按照成绩非递减排序  
(2)将排序好的数据转存到顺序表中  
(3)基于折半查找的方法对指定成绩的记录查找，输出查找成功的记录在有序序列中的位序及其完整记录内容或者说明查找失败。

实验原理:

先创建一个带表头的双向链表，然后进行核心部分：

。

1. 链表的快速排序：

对一趟排序来说，对前后两个指针，head为小， last为大，设置枢轴记录关键字（这里用了head指向的值），从last开始向前，当last指向的关键字小于枢轴记录关键字时，将其交换，然后用head向后，当head所指的关键字大于枢轴记录关键字时，将其交换，一直重复，直到head = last

1. 数组的对半查找：

用low high分别记下数组的开始与结束下标，然后当low不小于等于high时，说明没有找到，不然在中途找到mid=(low+high)/2 输出

**操作步骤：**

1. **按照提示，先输入需要输入的学生个数，然后回车**
2. **按照提示，依次输入学生需要，姓名，学号，然后回车**
3. **当输入完学生信息之后，会提示输入要查找的分数**
4. **按照提示输入要查找的分数，不需要查找时，输入-1**
5. **程序会自动输出该分数的所有学生信息，如没有该学生则会输出提示未找到**
6. **输入-1结束程序**

**算法思想：**

**核心算法为：**

1. **链表创建算法，为头插法，在类中定义一个公共属性为next与pre，类型为该类的指针，创建时用:**

void set\_data(student \*head) ; //将数据存储入链表中

**函数，调用类中的set\_属性 方法，来设置数据**

1. **链表的快速排序算法，用的函数有：**

void sort(student \*head , student \*last) ; //在链表中进行快速排序排序

student \*GetPartion(student \*head , student \*last) ; //快排时交换表中记录，使枢轴记录到位

void swap(student \*head , student \*last) ; //交换函数

**算法思路为：**对一趟排序来说，对前后两个指针，head为小， last为大，设置枢轴记录关键字（这里用了head指向的值），从last开始向前，当last指向的关键字小于枢轴记录关键字时，将其交换，然后用head向后，当head所指的关键字大于枢轴记录关键字时，将其交换，一直重复，直到head = last

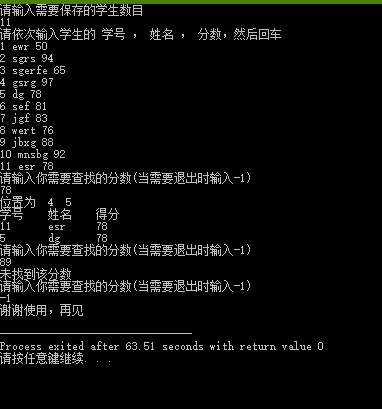
然后对head = last 位置的前后进行递归，（且这里注意判断条件）

1. **折半查找算法，用到的函数为：**

void search(student students[10000] , int n , int key) ; //折半查找

**将数组以及其大小传入函数中，然后初始化用low ,high 分别记下该数组的最大位置以及最小位置，用mid=(low+high)/2 来记下中间位置，然后对于查找值key>mid位置的值时，low = mid+1 , key<mid位置的值时，high=mid-1 , 而等于时，说明找到，进行输出(这里对相同连着的数据进行处理，比如如果有三个相连的相同数据，则都需要输出)，如果最后high>low，说明没找到。**

**测试截图：**

****

**问题体会：**

1. **一开始将next与pre设置为private，后来发现有些地方需要用的时候如果不设置一个get方法，难以调用，所以后来将其设置为public.**
2. **为了让程序模块化，所以所有的设置数据函数都在类中，用类的方法来实现，不过，在赋值数组时，发现之前的设置方法好像并不能用，所以在代码中设置了两组 set\_属性 函数，一组是不带参数的，创建链表时用，一组的带参数组，赋值数组时用**
3. **在快排时，对于递归条件没有考虑，（head||last != partion）导致一直越界，花了不少时间更改**
4. **在折半查找的输出中，对于分数相同的几个student的输出，进行优化，当找到时，由mid开始向左右展开，进而可以输出所有符合条件的人**

## 源代码：

#include<iostream>

#include<string.h>

#include<stdio.h>

#include<stdlib.h>

using namespace std ;

class student{ //定义一个学生类

private:

int id ; //学号属性

string name ; //姓名属性

int score ; //分数属性

public:

student \*next = NULL ; //用来指向下一个学生(链表)

student \*pre = NULL ; //用来指向前一个学生

void set\_id(){ //链表设置id的函数

cin>>id ;

}

void set\_name(){ //李彪设置name的函数

cin>>name ;

}

void set\_score(){ //链表设置分数

cin>>score ;

}

void set\_id2(int id2){ //数组中设置id的函数

this->id = id2 ;

}

void set\_name2(string name2){ //数组设置name的函数

this->name = name2 ;

}

void set\_score2(int score2){ //数组设置分数

this->score = score2 ;

}

int get\_id(){

return id ;

}

string get\_name(){

return name ;

}

int get\_score(){

return score ;

}

student(){ //构造函数

next = NULL ;

pre = NULL ;

}

};

void set\_data(student \*head) ; //将数据存储入链表中

void sort(student \*head , student \*last) ; //在链表中进行快速排序排序

student \*GetPartion(student \*head , student \*last) ; //快排时交换表中记录，使枢轴记录到位

void swap(student \*head , student \*last) ; //交换函数

void save(student \*head , student students[10000] , int n) ; //将链表中的数据储存在数组中

void search(student students[10000] , int n , int key) ; //折半查找

main(){

int n ; //用来保存需要输入的学生数目

cout<<"请输入需要保存的学生数目"<<endl ;

cin>>n ;

if(n == 0){

cout<<"不输入学生信息，程序退出" <<endl ;

exit(0) ;

}

student \*head ; //指向链表头结点的指针

student \*last , \*header ; //快速排序用的.

student students[n] ; //用来储存排好序的学生数组

head =new student() ;//创建链表头结点

cout<<"请依次输入学生的 学号 ， 姓名 ， 分数，然后回车"<<endl ;

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

set\_data(head) ;

}

header = head->next ;

last = header ;

while(last->next)

last = last->next ; //得到最后一个节点

sort(header , last) ; //对其排序

save(head , students , n) ; //将其储存在数组中

int order = 1 ; //输入指示分数

while(1){

cout<<"请输入你需要查找的分数(当需要退出时输入-1)"<<endl ;

cin>>order ; //输入分数

if(order!= -1){

search(students , n , order) ;

}else{

break ;

}

}

cout<<"谢谢使用，再见"<<endl ;

}

void set\_data(student \*head){ //设置数据的函数

student \*temp ;

temp = new student() ; //申请一个新的节点

temp->set\_id() ;

temp->set\_name() ;

temp->set\_score() ;

temp->next = head->next ;

head->next = temp ;

if(head->next->next){

head->next->next->pre = temp ;

}

temp = NULL ;

}

void swap(student \*head , student \*last){ //交换函数

int id = head->get\_id() ; //设置三个中间暂存变量

string name = head->get\_name() ;

int score = head->get\_score() ;

head->set\_id2(last->get\_id()) ;

head->set\_name2(last->get\_name()) ;

head->set\_score2(last->get\_score()) ;

last->set\_id2(id) ;

last->set\_name2(name) ;

last->set\_score2(score) ;

}

student \*GetPartion(student \*head , student \*last){ //快排时交换表中记录，使枢轴记录到位

int key = head->get\_score() ;

while(head != last){

while(head != last && last->get\_score() >= key){

last = last->pre ;

}

swap(head , last) ;

while(head != last && head->get\_score() <= key){

head = head->next ;

}

swap(head , last) ;

}

return head ;

}

void sort(student \*head , student \*last) {//在链表中进行快速排序

if(head != last)

{

student \*partion = GetPartion(head , last);

if(partion != head)

sort(head , partion->pre);

if(partion != last)

sort(partion->next , last);

}

}

void save(student \*head , student students[10000] , int n) { //将链表中的数据储存在数组中

student \*p = head->next ;

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

students[i].set\_id2(p->get\_id()) ;

students[i].set\_name2(p->get\_name()) ;

students[i].set\_score2(p->get\_score()) ;

p = p->next ;

}

}

void search(student students[10000] , int n , int key) { //折半查找

int low = 0, high = n-1 , mid ;

while(low <= high){

mid = ((low + high)/2) ;

if(key < students[mid].get\_score()){

high = mid - 1 ;

}

else if(key > students[mid].get\_score()){

low = mid + 1 ;

}

else{ //下面都是找到了然后输出的代码

cout<<"位置为 "<<mid+1<<" " ; //表示找到了该位置

int temp1 = mid-1 , temp2 = mid + 1 ;

if(temp1<n && temp1>=0 && students[temp1].get\_score()==key)

cout<<temp1+1<<" " ;

if(temp2<n && temp2>=0 && students[temp2].get\_score()==key)

cout<<temp2+1<<" " ;

cout<<endl ;

cout<<"学号\t姓名\t得分\t"<<endl ;

cout<<students[mid].get\_id()<<"\t"<<students[mid].get\_name()<<"\t"<<students[mid].get\_score()<<endl ;

if(temp1<n && temp1>=0 && students[temp1].get\_score()==key)

cout<<students[temp1].get\_id()<<"\t"<<students[temp1].get\_name()<<"\t"<<students[temp1].get\_score()<<endl ;

if(temp2<n && temp2>=0 && students[temp2].get\_score()==key)

cout<<students[temp2].get\_id()<<"\t"<<students[temp2].get\_name()<<"\t"<<students[temp2].get\_score()<<endl ;

return ;

}

}

cout<<"未找到该分数"<<endl ; //循环结束，且没有找到

}