**实验9 11-26oj**

1. 实验的目的和要求

学习c++编程的原理和方法

1. 实验内容
2. 实验准备

C++中创建结构体

C++中结构体的使用

C++中链表结构的使用，包括链表的结点生成，搜索，链接，删除等一系列操作

C++中通过将链表作为函数的返回值从而实现使用函数对链表进行操作

(二)实验项目

本次实验四道题目都是基于一系列基本功能，故不分成四道题目而是整合成一整道题目从而反映这次试验中出现的问题。

题目描述

定义一个学生成绩结构体类型，包含“学号”（三位数）、“姓名”（不超过20个字符）、“性别”、“年龄”、“班级”(一位数字)、“英语”、“数学”、“物理”、“总分”（计算得出）、“名次”（计算得出）等信息。   
定义一个用于存储“学生成绩结构体”的链表，包含6个子函数分别实现下述功能：    
print 显示链表，将链表所有内容输出，参考函数形式：void print(const node\* head)  
add  添加结点, 将学生插入链表尾部，参考函数形式：node\* insStudent(node\* head, node\* p) 输入示例：add 001 Jack male 18 1 60 80 100  
del 删除结点, 删除指定学号的学生信息，参考函数形式：node\* delStudent(node\* head, int sno) 输入示例：del 002  
rmL 查找总分最低成绩学生，将其删除，参考函数形式： node\* rmLastStudent(node\* head)   
  
sortE 按英语成绩从大到小排序, 参考函数形式：node\* sortByEnglish(node\* head) //测试用例中英语分数各不相同   
sortR 按名次从第一名到最后一名排序，参考函数形式：node\* sortByRank(node\* head) //测试用例中总分各不相同   
reverse 逆序排序， 参考函数形式： node\* reverse(node\* head)

insert 插入结点, 将学生插入链表头部，参考函数形式：node\* insStudent(node\* head, node\* p) 输入示例：insert 001 Jack male 18 1 60 80 100

分析

首先初始化内容就有很多问题

1结构体的创建，根据题目要求struct来创建一个保存学生信息的结构体

2链表的创建，链表创建格式有必要在这里重申一下

struct node{

student data;

node \*next;

};

可以看到这是一个简单的单向链表

单向链表每一个结点有两个部分组成第一个部分用于保存数据，另一个则是指向下一个节点的指针，该指针的指向类型则为链表的这个类型。

3读入数据，首先操作符的判断使用strcmp函数最为保险，其次则是如果不使用重载那么只能将student的每个信息读入，同时注意如何创建新的结点，node\*p=new node();p->next=NULL;然后读入数据要注意是cin<<p->data.学生信息。

4构造空链表node\*head=new node();head=NULL;

接下来则是不同的操作函数

首先我们从最基本的三个操作函数开始

Add,insert和print函数

Add

node\* addStudent(node\* head, node\* p)

{

if(head==NULL)

return p;

head->next=addStudent(head->next,p);

return head;

}

可以看到add这个操作使用递归来实现，如果当前结点是空的那么就返回即可，否则再向后递归

Insert

node\* insStudent(node\* head, node\* p)

{

node \*p0=new node();

p0=head;

p->next=head;

return p;

}

Ins函数则更加简单，直接创建一个新的结点，把先前链表添加在后面即可

Print函数这里就不粘帖代码了，简单的分析一下，还是从头指针开始，只要当前指针不是空指针就将其中的data输出，然后递归，在print当前结点之后的链表内容就可以了。由于题目要求如果当前链表没有学生信息的内容是要输出print的因而在主调函数中我们再加入一个布尔类型的变量从而实现该功能。

除了上述的一些操作，容易被忽视的就是我们需要计算学生的排名，我使用的是在将结点加入到链表中的时候就计算出该学生的排名来，在该结点还没有进入链表中的时候，先默认学生的排名是1，然后从链表的头结点开始进行遍历，如果p的总分高于当前结点，那么当前结点名次++，如果低于那么就将p->data.rank++

接下来是del和rmL函数，为了能够更加简便的使用del函数我们这里将rmL的返回值定义为int变量

因而rmL函数的逻辑非常简单直接从头结点开始遍历逐一比较maxrank和当前结点所指向的data中rank的大小，找到那个学生并且返回他的学号即可。

Del函数使用递归，如果当前结点中所指向的学生的学号就是最低的分数，那么就直接返回该结点之后的链表，否则的话在del（head->next,sno）即可。

}

node\* delStudent(node\* head, int sno)

{

if(head!=NULL)

{if(head->data.num==sno)return (head->next);

else head->next=delStudent(head->next,sno);}

return (head);

}

最后需要实现的是两个排序和逆序实际上函数的逻辑是相同的因此只要分析一个就可以了

node\* sortByEnglish(node\* head)

{

node\*head0=new node();

head0=head;

while(head!=NULL)

{

node\*p1=new node();

p1=head;

while(p1!=NULL)

{

if(p1->data.english>head->data.english)

{

student temp;

temp=p1->data;

p1->data=head->data;

head->data=temp;

}

p1=p1->next;

}

head=head->next;

}

return head0;

}

利用的思想完全仿照直接插入排序，从链表的头结点开始逐一遍历，然后再分别和之后的结点对比，一但满足交换条件那么就交换所含内容（data）,运行次数是链表长度的两倍。逆序排序的逻辑完全相同，甚至都不需要比较交换条件，只要每次直接交换就可以了。

源代码

源代码内容过多超过一百行就不在此黏贴并且大部分函数在分析中已经作了解释

测试数据

Input

17

add 001 Jack male 18 1 60 80 100

add 002 Mary female 18 2 98 99 98

add 003 Mike male 18 1 98 99 100

add 004 Jerry female 18 2 100 99 100

add 005 Bob male 18 1 60 80 90

add 006 Justin female 18 2 98 89 98

print

sortE

print

sortR

print

reverse

print

rmL

rmL

rmL

print

Output

001 Jack male 18 1 60 80 100 240 5

002 Mary female 18 2 98 99 98 295 3

003 Mike male 18 1 98 99 100 297 2

004 Jerry female 18 2 100 99 100 299 1

005 Bob male 18 1 60 80 90 230 6

006 Justin female 18 2 98 89 98 285 4

004 Jerry female 18 2 100 99 100 299 1

003 Mike male 18 1 98 99 100 297 2

002 Mary female 18 2 98 99 98 295 3

006 Justin female 18 2 98 89 98 285 4

005 Bob male 18 1 60 80 90 230 6

001 Jack male 18 1 60 80 100 240 5

004 Jerry female 18 2 100 99 100 299 1

003 Mike male 18 1 98 99 100 297 2

002 Mary female 18 2 98 99 98 295 3

006 Justin female 18 2 98 89 98 285 4

001 Jack male 18 1 60 80 100 240 5

005 Bob male 18 1 60 80 90 230 6

005 Bob male 18 1 60 80 90 230 6

001 Jack male 18 1 60 80 100 240 5

006 Justin female 18 2 98 89 98 285 4

002 Mary female 18 2 98 99 98 295 3

003 Mike male 18 1 98 99 100 297 2

004 Jerry female 18 2 100 99 100 299 1

005 Bob male 18 1 60 80 90 230 6

001 Jack male 18 1 60 80 100 240 5

006 Justin female 18 2 98 89 98 285 4

002 Mary female 18 2 98 99 98 295 3

003 Mike male 18 1 98 99 100 297 2

004 Jerry female 18 2 100 99 100 299 1

1. 小结

这次实验难度非常大（然鹅还是有巨佬30min做完很想看看他的实验报告233），主要学习并且实践了如何对链表进行操作，如何有效的将结构体指针（链表）作为函数的返回值从而实现一系列的功能。在链表的处理过程中递归的思想有的时候可以有效地简化或者将问题的处理变得更加有条理。处理指针问题要注意的一个地方就是需要注意到空指针问题，如果无法有效的处理空指针而直接想要访问里面的数据则会造成内存越界，因而要要注意到这一个细节。其他的问题在处理链表问题的时候，主要围绕两个部分如何链接和断开链接，这样才可以形成完整的链表。并且也要记住链表是一整块链而不是离散的，所以每一步的操作除非遇到了空指针否则都要有链接，不然很容易将链表消亡或者变成全空。对于排序操作实际上可以使用递归思想，从头结点(并非链表头结点，只是当前函数中传入的形参)开始遍历对于每一个结点，如果比较值>0那么就排序当前头结点之前，如果是<=0那么比较之后的内容就可以了。对于链表的问题还有许多要处理的地方，如果要更进一步掌握的话，还需要更多的操练。