《C语言程序设计实训2》报告

设 计 题 目： 基于链表的学生信息管理系统

学 院 名 称： 信息科学技术学院

专 业： 网络工程

班 级： 二班

姓 名： 李毓琪 学 号：2220191195

指 导 教 师： 张海昕

提 交 日 期： 2020年7月20日

**一、 实验内容**：

编写并调试程序，实现学校各专业班级学生信息的管理。要求一个工程下多个文件。main函数以单独的文件main.cpp存放。在头文件student.h中定义学生信息的结构体类型，包括：学号、姓名、专业、班级、3门成绩；和符号常量N（学生数）。（同一班级的学生可以属于不同的专业，同一专业的学生可以属于不同的班级）。

全局类型的定义如下：

#define N 12

struct Student{

char num[15];//学号

char name[15];//姓名

char major[10];//专业（computer,software,network）

int classNo;//班级(1-2)

int score[3];//3门课的成绩(0-2)

struct Student \*next;//指向下一结点的指针

};

typedef struct Student STU;

**二、实验要求：**

1. main函数：以菜单形式将各项功能提供给用户，根据用户的选择，调用相应的函数。

**请输入功能序号完成相应功能**

1.从文件中读取学生信息，并显示链表中各个学生的信息

2.查找指定的学生信息，并修改

3.求总分最高的同学，并输出其信息

4.对学生信息按平均成绩进行排序

5.输入一个学生的信息，按学号顺序插入到相应位置

6.删除指定的学生信息

7.查找某个专业的，某门课程的成绩小于某个分数的学生

8.删除某个班级的，某门课程的成绩小于某个分数的学生

9.将更新后的学生信息保存到另一个文件中

其他，退出

1. 定义函数STU \*CreateList（char \* filename）：从二进制文件“studentInfo.dat”中依次读取12个学生的数据(在文件中学生信息按学号由低到高依次存放)，建立**带头结点**（特殊结点，不放学生信息）的、有序的单链表(链表中各学生信息按学号由低到高依次存放), 返回表头指针。每次从文件中读一个学生的信息，并将该学生信息插入到链表的表尾。设置头结点的目的：在链表中第一个学生结点之前插入或者是删除第一个学生结点的时候不必做特殊处理）

更灵活的方法：从文件中循环读数据时，根据当前学生信息的next成员的值，决定是否进行下次循环，如果next==NULL，则表明读到最后一个学生信息，循环结束。

1. 定义函数void Output (STU \*head)：以指向链表头结点的指针为参数，将链表中所有学生的信息表格化输出。

定义函数 void OutOneNode(STU \*p): 输出参数所指的链表中一个结点的信息。

1. STU \*Max(STU \*head)：查找总成绩最高的学生，返回指向该学生结点的指针。

考虑特殊情况，找到两个及以上相同最高分同学的情况，如何处理？

1. 定义函数STU \*Search\_num(STU \* head ，char \*num )：查找指定的学生，返回指向该学生结点的指针。可以按照学号或者姓名查找，然后可以修改该同学的所有数据
2. 定义函数 void InsertList(STU \* head, STU stu )：将某个学生信息插入到链表中的相应位置，并保持此链表按学号的有序性。

红色表示新插入的学生学号2000 2002 2008 2012 2017 2020 2100 2200 2205 2220…

1. 定义函数void Delete\_num(STU \* head，char \*num)：从链表中删除指定学号或姓名的学生。
2. 定义函数STU \*Search\_major \_subject\_score(STU \*head, char \*major, int subject , int grade )：查找某个专业的、某门课程的成绩小于某个分数的学生，返回指向该学生结点的指针。

若假设只找到一个学生即可，则返回指向该学生的指针；若假设如果有多个学生都要找到，则需另建立一个链表，从堆中申请空间，将所有满足条件的学生信息拷贝过来，链接起来（不破坏原来的学生信息链表），返回该指向链表头的指针。

可以将查询到的结果另存为新文件

1. 定义函数void Delete\_ class \_subject\_score(STU \*head, int classno, int subject, int grade)：从链表中删除某个班级的、某门课程的成绩小于某个分数的学生。

两种方法：在原链表修改，或者新建一个链表，

操作后要能够另存为新文件。

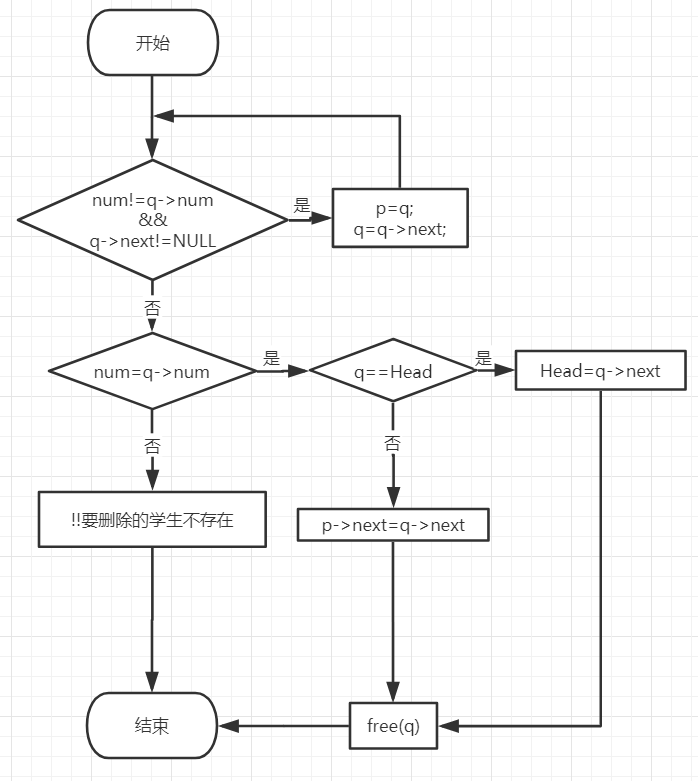
1. 定义函数Save(STU \*head)：将更新后的学生信息存入一个新文件中，例如名字为“studentInfoNew.dat”的文件。存放到外存时，不用指定长度。对链表的每一个节点操作，直到尾节点。\*next的值是NULL
2. 对学生信息按平均成绩进行排序。自定义函数名，排序方法不限。

可以不做这个功能，这个是加分项

**要求：**除main函数和Output、OutOneNode函数外，其它函数中不允许有键盘输入和屏幕输出的操作。

**三、算法流程图**

函数Delete\_ class \_subject\_score的算法流程图

****

**四、程序清单**

Main.cpp主函数部分：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include"function.h"

#define N 12

typedef struct Student STU;

void InsertList(STU \* head, STU stu );//定义插入函数

void Delete\_num(STU \* head,char \*num); //定义删除指定学号学生信息的函数

void Save(char \*filename); //定义储存函数

STU \*CreateList(char \*filename);//定义输入函数

STU \*Search\_num(STU \*head ,char \*num ); //定义查找指定学号学生信息函数

void Output(STU \*head); //格式化输出函数

STU \*Max(STU \*head); //获取单科成绩最高学生的信息

STU \*Sort(STU \*head);//起泡法排序

void OutOneNode(STU \*p);//格式化输出指针指向的结构体内容

int main(){ //主函数

int choice;//选择变量

STU \*head; //头函数

label: do{

printf("╭───────────────────────────────────────────────────────╮\n");

printf("│\t请输入对应序号进入对应功能\n");

printf("│1. 从文件中读取学生信息，并显示链表中各个学生的信息\n");

printf("│2. 查找指定学号的学生，并修改 \n");

printf("│3. 求总分最高的同学，并输出其信息 \n");

printf("│4. 对所有学生的总成绩进行排序 \n");

printf("│5. 输入一个学生的信息，按学号顺序插入到相应位置 \n");

printf("│6. 删除指定的学生信息 \n");

printf("│7. 查找某个专业的，某门课程的成绩小于某个分数的学生 \n");

printf("│8. 删除某个班级的，某门课程的成绩小于某个分数的学生 \n");

printf("│9. 更新后的学生信息保存到另一个文件中\n");

printf("│0. 格式化显示所有学生的信息 \n");

printf("│其他.退出 \n");

printf("╰───────────────────────────────────────────────────────╯\n");

printf(" 请选择项目 :");

scanf("%d",&choice);

printf("─────────────────────────────────────────────────────────\n");

getchar();

switch(choice){

case 0:

Output(Head);

printf("─────────────────────────────────────────────────────────\n");

printf("按Enter键返回主菜单");

getchar();

break;

case 1:

char Filename[20];

printf("请输入需要打开的文件名:");

scanf("%s",&Filename);//键入需要打开文件名称

printf("─────────────────────────────────────────────────────────\n");

CreateList(Filename);//输入学生信息

printf("输入成功！按Enter键返回主菜单");

getchar();getchar();

break;

case 2:

STU \*p;//定义指针p

int chio;

char fixnum[15],fixname[15];

printf("请选择查询的类型：1.学号 2姓名:");

scanf("%d",&chio);

getchar();

switch(chio){

case 1:

printf("请输入学号:");

gets(fixnum);

break;

case 2:

printf("请输入姓名:");

gets(fixname);

p=Head;

while(p){

if(strcmp(fixname,p->name)==0){

strcpy(fixnum,p->num);

break;

}

else

p=p->next;

}

break;

default :

printf("!!请输入正确的序号,按Enter键返回主菜单\n");

getchar();

goto label;

}

p=Search\_num(head,fixnum);//p接收search\_num函数返回的指针指向需要修改的学生信息

if(p==NULL){

printf("，按Enter键返回主菜单");

getchar();

goto label;

}

int ch;//选择函数

printf("请选择需要修改的信息：\n1.学号\t2.姓名\t3.专业\t4.班级\t5.分数:");

scanf("%d",&ch);//生成列表供用户选择

getchar();

switch(ch){

case 1:

printf("请输入学号:");

gets(p->num);

break;

case 2:

printf("请输入姓名:");

gets(p->name);

break;

case 3:

printf("请输入专业:");

gets(p->major);

break;

case 4:

printf("请输入班级:");

scanf("%d",&p->classNo);

break;

case 5:

printf("请选择学科：（1.高数 2.线代 3.C语言）:");

int chs;//选择函数

scanf("%d",&chs);

printf("请输入分数:");

scanf("%d",&p->score[chs-1]);

break;

default:

printf("！！输入错误，按Enter键返回主菜单");

break;break;

}

printf("修改成功！！！" );

getchar();

break;

case 3:

STU \*q;//定义结构体指针

q=Max(head);//指针指向总成绩最高的学生

printf("成绩最高的学生的信息为：\n");//

OutOneNode(q);//输出该指针指向的结构体

printf("按Enter键返回主菜单");

getchar();

break;

case 4:

struct Student \*l;

l=new Student;

l=Sort(Head);

Output(l);

printf("排序成功！按Enter键返回主菜单");

Head=l;//避免部分输出

getchar();

break;

case 5:

STU stu;//定义需要插入的学生信息

printf("请输入新增学生的学号:");

gets(stu.num);

printf("请输入新增学生的姓名:");

gets(stu.name);

printf("请输入新增学生的专业:");

gets(stu.major);

printf("请输入新增学生的班级:");

scanf("%d",&stu.classNo);

printf("请输入新增学生的三门课程成绩（高数 线代 C语言 ▲中间用空格隔开▼）");

scanf("%d %d %d",&stu.score[0],&stu.score[1],&stu.score[2]);

InsertList(head,stu);//调用函数插入学生信息

getchar();

break;

case 6:

char num[15];

printf(" 请输入要删除的学生的学号 :");

gets(num); //输入学号

Delete\_num(head,num);//删除学生信息

printf("删除成功!按Enter键返回主菜单");

getchar();

break;

case 7:

char majorNo[10];

int lessonNo,score;

p=Head;//指向头函数

printf("请输入需要查找的专业(network software computer):") ;

gets(majorNo);

printf("请输入需要查找的课程(1=高数 2=线代 3=C语言)");

scanf("%d",&lessonNo);

printf("请输入需要筛选的学生成绩小于多少:");

scanf("%d",&score);

struct Student \*z;

z=new Student;

while(p){

if(strcmp(p->major,majorNo)!=0||p->score[lessonNo-1]>score){//寻找专业符合的学生

Delete\_num(head,p->num);

count--;

}

p=p->next;

}

printf("\n筛选成功！按Enter键返回主菜单%d\n",count);

getchar();

break;

case 8:

p=Head; //头函数

int ClassN,LessonN,Score;

printf("请输入所要删除的班级：");

scanf("%d",&ClassN);

printf("请输入需要查找的课程(1=高数 2=线代 3=C语言)");

scanf("%d",&LessonN);

printf("请输入需要筛选的学生成绩小于多少:");

scanf("%d",&Score);

while(p){

if(Score>p->score[LessonN-1]&&p->classNo==ClassN){//寻找符合分数和班级的学生

Delete\_num(head,p->num);//找到学生后对其进行删除

count--;

}

p=p->next; //指向下一个结构体

}

printf("删除成功,按Enter键返回主菜单");

getchar();getchar();

break;

case 9:

char filename[20];

printf("请输入需要保存的文件名");

scanf("%s",&filename);

Save(filename);//保存函数

getchar();

break;

default:

exit(0);

break;

}

}while(1);

return 0;

}

Function.h函数部分：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define N 12

struct Student{//链表结构

char num[15];//学号

char name[15];//姓名

char major[10];//专业（computer,software,network）

int classNo;//班级(1-2)

int score[3];//3门课的成绩(0-2)

struct Student \*next;//指向下一结点的指针

};

typedef struct Student STU;

struct Student \*Head; //定义头文件Head

int count=0; //定义计数函数

void Output(STU \*head){ //格式化输出函数

struct Student \*p;//声明一个指针p

p=head; //p指向头函数

while(p){

printf("学号：%-15s 姓名：%-15s 专业%-10s 班级：%d ",p->num,p->name,p->major,p->classNo);

printf("高数：%d 线代：%d C语言:%d 总成绩：%d\n",p->score[0],p->score[1],p->score[2],(p->score[0]+p->score[1]+p->score[2]));

p=p->next; //指向下一个结构体

}

}

void InsertList(STU \* head, STU stu ){ //插入学生信息

struct Student \*p,\*q;

p=Head; //指向头函数

q=(struct Student \*)malloc(sizeof(struct Student)); //为q分配地址空间

\*q=stu;//将stu的内容赋予p

while(p){

if(strcmp(p->next->num,stu.num)>=0){//按学号次序插入学生的讯息

q->next=p->next;

p->next=q; //进行插入

break;

}

p=p->next;

}

}

STU \*Search\_num(STU \*head ,char \*num ){ //搜索指定学号的学生信息

struct Student \*p;

p=Head;//指向头函数

while(p){

if(strcmp(p->num,num)==0){//查找学号相同的学生

return p; //返回P的指针

}

p=p->next;

}

if(p==NULL)

printf(" !!查无此人");

return NULL;

}

void Delete\_num(STU \* head,char \*num){

struct Student \*p,\*q;

q=Head;

while(strcmp(num,q->num)!=0&&q->next!=NULL){

p=q;

q=q->next;

}

if(strcmp(num,q->num)==0){

if(q==Head)

Head=q->next;

else

p->next=q->next;

free(q);

return;

}

else{

printf("\n!!要删除的学生不存在\n");

return;

}

}

STU \*Max(STU \*head){//查找总成绩最高的学生，并返回指针

struct Student \*p,\*q;

int ch;

p=Head;//指向头函数

q=new Student;//分配空间

q->score[0]= q->score[1]=q->score[2]=0;

while(p){

if(p->score[0]+p->score[1]+p->score[2]>q->score[0]+q->score[1]+q->score[2])

q=p; //对链表进行依次比较，查找到最大的成绩，将其内容赋予q

p=p->next;

}

return q; //返回q的指针

}

void Save(char \*filename){ //保存学生信息

FILE \*fp;

struct Student \*p=Head; //定义结构体指针，并指向头函数

if((fp=fopen(filename,"wb"))==NULL){

printf(" 文件打开失败 !");

return;

}

if(count){

fprintf(fp,"%d\n",count);

while(p){

fprintf(fp,"%s %s %s %d %d %d %d\n",p->num,p->name,p->major,p->classNo,p->score[0],p->score[1],p->score[2]);

p=p->next;

}

}

else{

printf("！！内存中没有学生");

getchar(); getchar();

return;

}

if(fclose(fp))

printf(" 文件关闭失败 !");

else

printf("文件保存成功！");

getchar();

}

STU \*CreateList(char \*filename){ //输入学生信息

int i;

FILE \*fp;

struct Student \*p,\*q; // 定义结构体指针

if((fp=fopen(filename,"rb"))==NULL){ //打开文件

printf("!!文件打开失败,按Enter键返回主菜单");

getchar();

return 0;

}

p=Head; //指向头函数

while(p){

q=p->next; //生成链表

free(p); //释放空间

p=q;

}

fscanf(fp,"%d",&count); //进行计数赋予

for(i=0;i<count;i++){

p=(struct Student \*)malloc(sizeof(struct Student));//分配空间地址

fscanf(fp,"%s %s %s %d %d %d %d\n",&p->num,p->name,&p->major,&p->classNo,&p->score[0],&p->score[1],&p->score[2]); //进行赋值

p->next=NULL; //尾节点=NULL

if(i==0)

Head=p;//单个节点时头函数即为p

else

q->next=p;

q=p;

}

if(fclose(fp))

printf("!!!文件关闭失败，按Enter键返回主菜单");

}

STU \*Sort(STU \*head){

if (!head||!head->next) {

return head;

}

struct Student\* O= NULL;//原链表

struct Student\* h= new Student;//新链表，并且分配空间

struct Student\* a= NULL;//返回指针链表

h->next=head;//原链表再加一个头函数

head=head->next;

h->next->next=NULL;

while(head){//检查头函数

O=head;//O赋值head

head=head->next;//head指向下一项

O->next=NULL;//O为末指针，下一项指向NULL

struct Student\* p=h->next;

struct Student\* t=h;

while (p!=NULL&&p->score[0]+p->score[1]+p->score[2]<=O->score[0]+O->score[1]+O->score[2]) {

t=p;//进行赋值

p=p->next;//进行查找

}

O->next=t->next;

t->next=O;//插入

}

a=h->next;//生成目标链表

return a;

}

void OutOneNode(STU \*p){//格式化输出一个学生的信息

printf("学号：%-15s 姓名：%-15s 专业%-10s 班级：%d",p->num,p->name,p->major,p->classNo);

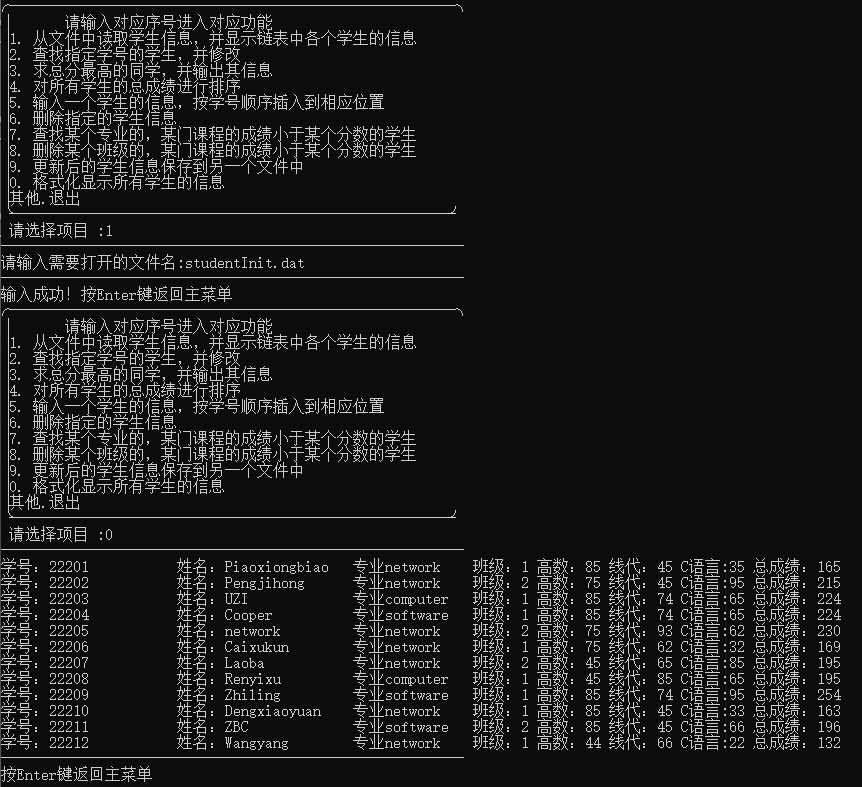
printf("高数：%d 线代：%d C语言:%d 总成绩：%d\n",p->score[0],p->score[1],p->score[2],(p->score[0]+p->score[1]+p->score[2]));

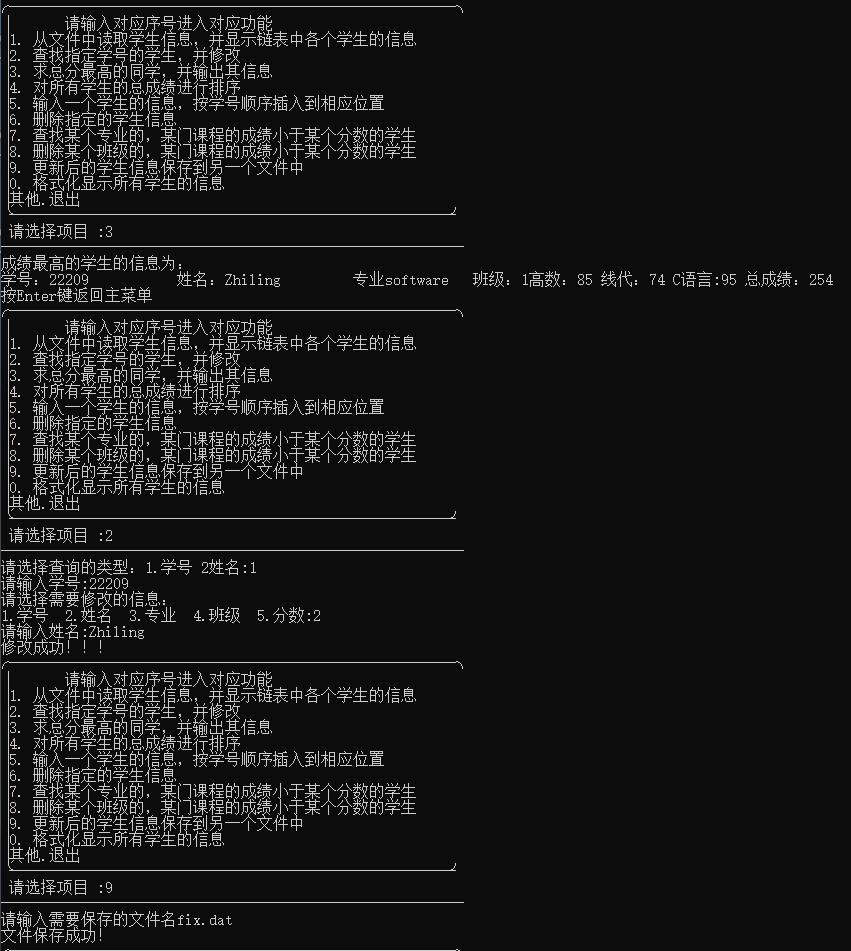
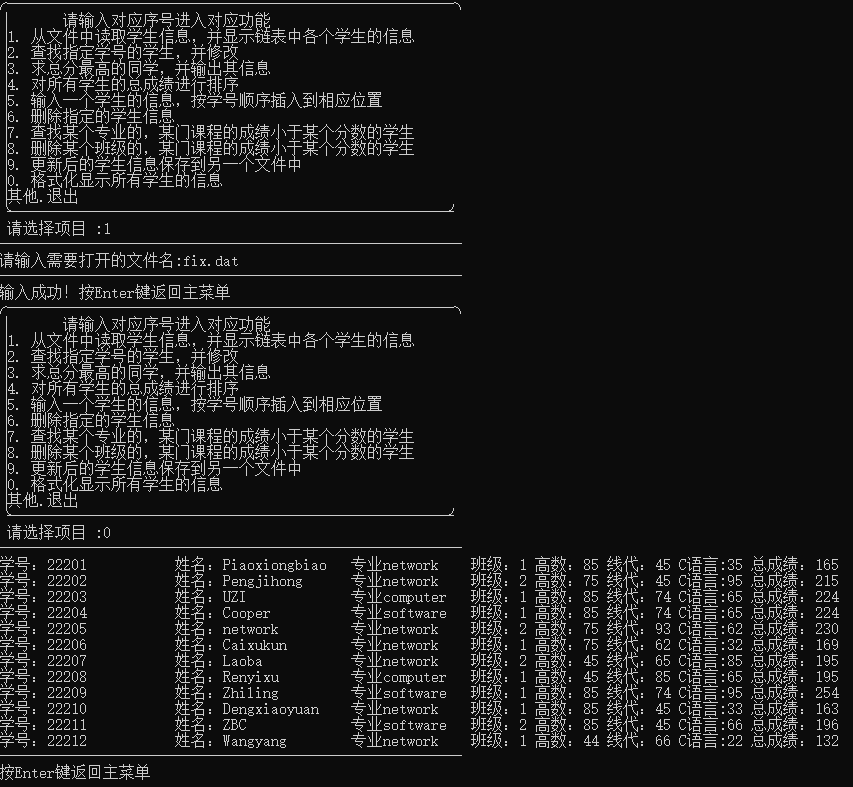
}

**五、程序测试**

测试用例1：查找最高分  
1、打开表studentInit.dat  
2、显示所有人员信息  
3、查找最高分的同学

4、修改最高分同学的姓名为Zhiling  
5、保存链表信息到新文件fix.dat  
6、打开新文件查看

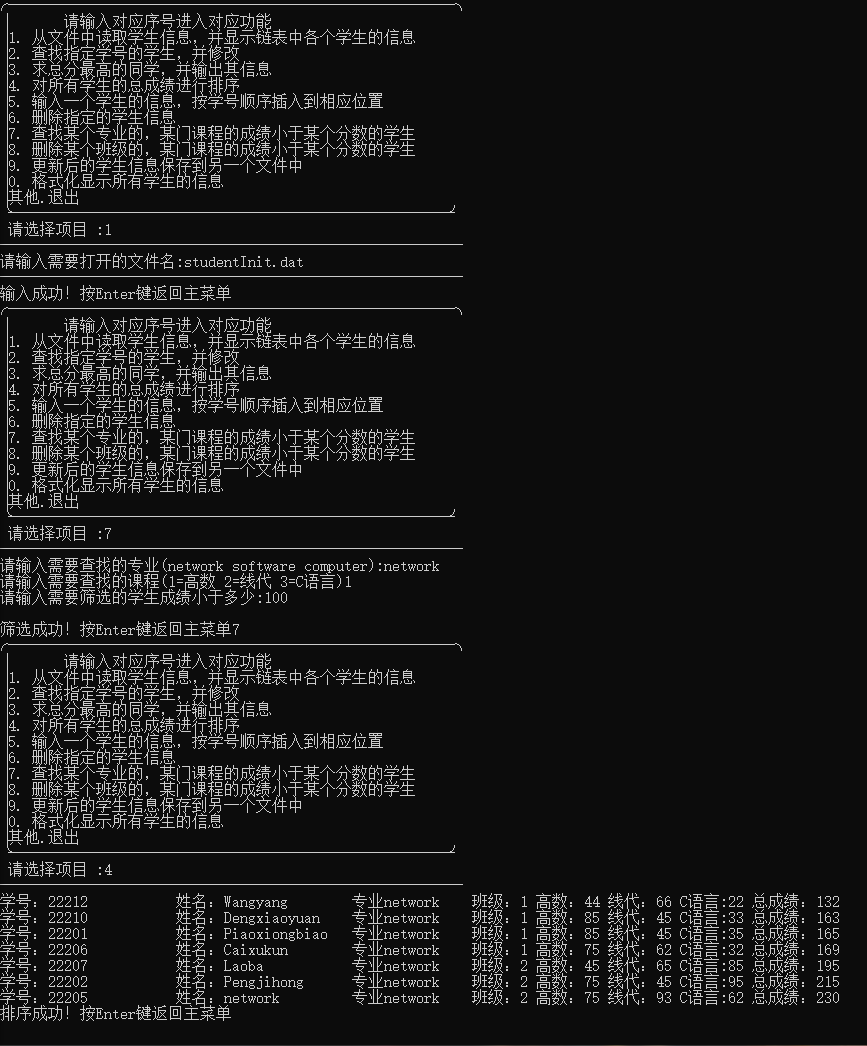




测试用例2：将network专业学生信息按总成绩进行排序

1. 打开表studentInit.dat
2. 筛选network专业高数成绩低于100分的学生

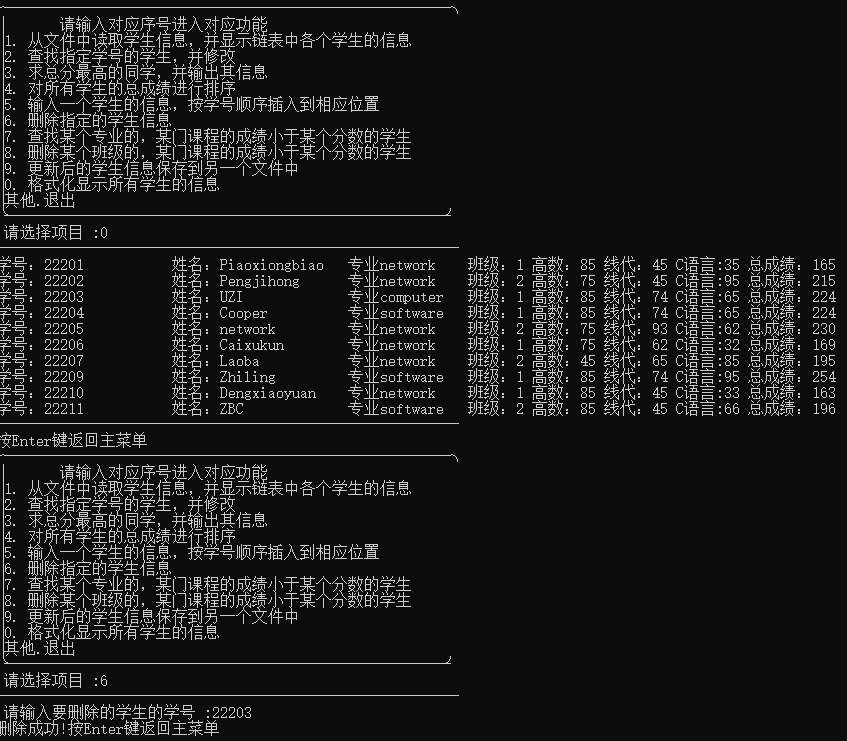
3、将筛选后的学生按照总成绩进行排序并显示排序后所有人的信息

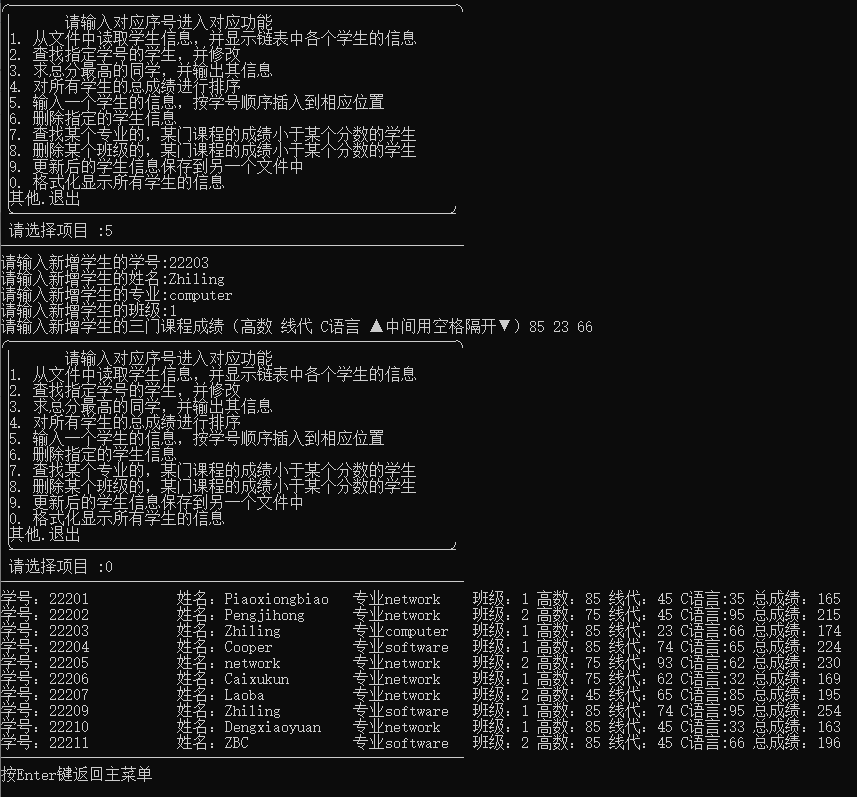


测试用例3：删除1班高数成绩小于60分的学生信息之后对学号为22203的学生的信息进行删除并插入处理

1. 打开表（studentInit.dat）
2. 删除1班 高数成绩 低于60分的学生
3. 显示删除之后人员信息
4. 删除学号为22203的学生
5. 插入学号22203 姓名Zhiling 专业computer 班级1 成绩85 23 66的学生
6. 显示修改之后的人员信息





****

1. **实验总结**

遇到的问题:

1. 指针指向错误，原因有未给指针分配地址导致指针指向错误，或是循环出现错误。
2. 打开删除后的文档时，会输出12个人的数据，空位置的学生信息输出乱码。解决方法，再删除信息同时把计数函数的值减1，使保存时的count等于学生信息数。
3. 编写起泡法排序后的数据只有部分数据位置发生变化，且未正常排序。解决方法，更换成插入排序方法，更加简单便捷。
4. 再分配空间时可以用malloc也可以用new，使用new时无需设置地址空间大小。

数组与链表的异同：

数组：在内存中，数组是一块连续的区域，因为时连续的，查找时可直接进行查找，但是进行插入和删除处理时比较麻烦，而且因为空间限制，当数组空间不足时智能重新定义数组。

链表：在内存中，链表不一定时连续的区域，每个节点都保存着下一个节点的指针只想下一个节点。进行插入和删除处理时很容易，只需要修改指针即可。但是查找的效率比较低，因为需要从头开始遍历。

在实训课程中掌握了链表和数组的结构，掌握简单排序，起泡法排序，直接插入排序方法，并能够在链表中和数组结构中实现。增强动手能力，和对C语言的理解。能够在繁杂枯燥的代码中找到乐趣，熟悉了C语言开发环境，并且能够新建函数文件，主函数能够调用函数文件中函数等操作。能够把bug的挫折转化为前进的动力，并且能够通过网上查询资料等方式进行学习更多的知识。