# 目录

[目录 1](#_Toc33114617)

[一．需求分析 2](#_Toc33114618)

[二．总体设计 2](#_Toc33114619)

[三．详细设计 3](#_Toc33114620)

[3.1编写目的 3](#_Toc33114621)

[3.2详细设计 3](#_Toc33114623)

[四．调试与运行 3](#_Toc33114624)

[五．源程序清单 8](#_Toc33114625)

一．需求分析

**1.1 编写目的**：通过对用户需求进行调查分析，写出需求分析文档作为项目设计的基本准则要求，以及作为系统分析员进行系统分析和测试人员进行测试时的手册

**1.2需求概述：**  设计一个学生成绩排名系统，使之能提供以下功能：  录入学生信息并保存  浏览学生信息  查询学生信息  修改学生信息并保存  添加学生信息  统计计算学生信息

**1.3 需求说明：**本系统所实现的主要功能：

（1）添加新成员，实现对学生信息的录入。

（2）删除学生信息。需要删除学生信息时，释放掉它所占据的数据库空间，就可以在列表中选中所要删除的记录，然后点击删除如果确认删除则选择确定选项即可删除所选的记录。

（3）修改学生信息，由于某种原因可能导致在记录学生信息时出现错误，需对学生信息进行修改。在学生信息系统主窗口中选择所要修改的成员信息进行修改。

（4）显示学生信息，在删除学生信息和添加学生信息成功后，输入显示选项则可对学生信息记录进行更新，显示学生信息的最新记录。

（5）姓名检索信息，可以显示出当前那一条或几条学生信息，用名字来查看学生信息。

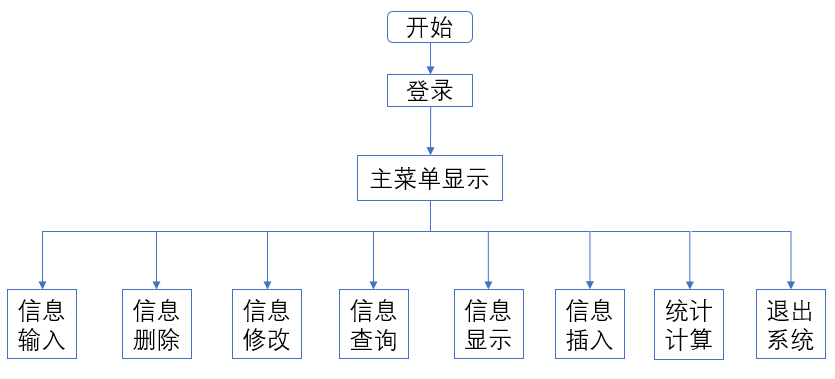
(6)对学生成绩进行排序，总分计算等统计功能。

综上本系统可以对学生成绩信息实现科学自动的管理，其中包括显示学生基本信息，并实现学生信息的添加、删除、显示、统计及检索。经过调试，程序性能良好。

二．总体设计

总体功能部分设计图如下，具体主菜单功能选项包括：

1. 学生信息输入
2. 学生信息输出显示
3. 课程设置
4. 学生信息添加
5. 学生信息查找
6. 学生信息删除
7. 学生信息修改
8. 学生成绩信息排序
9. 学生信息统计计算
10. 退出



三．详细设计

3.1编写目的

根据项目概要设计说明书，在项目概要设计的基础之上，对系统进行详细设计，以便下一步进行程序编码工作。

3.2详细设计

1．课程结构体

包括课程名称和课程成绩。

1. 课程 类

私有成员包括：课程结构体数组，课程数量，成绩总和，平均成绩

公有成员包括：总分，平均分计算，课程信息显示，增加课程，删除课程，成绩的输入与输出显示等。

1. 学生 类

私有成员包括：学生的姓名，学号，班级，课程等

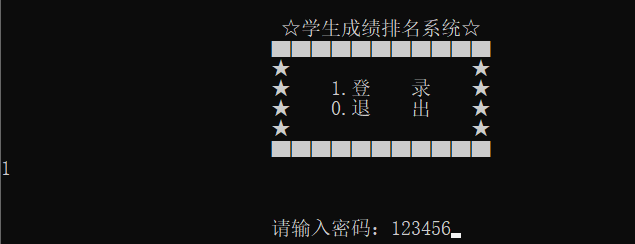
公有成员包括：数据的输入与输出

1. 学生的信息链表 类

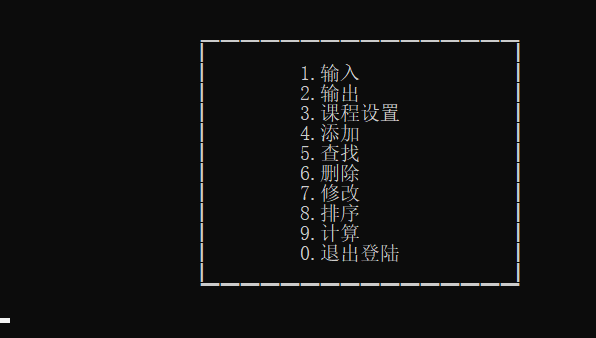
主要进行链表的操作，查找，删除，修改，排序，统计计算与输出显示

# 四．调试与运行

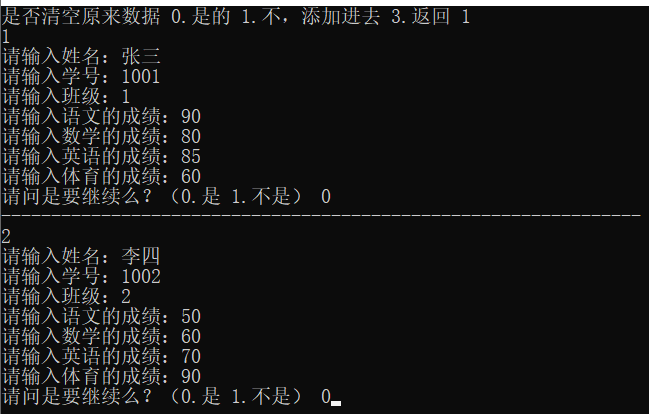
登录界面



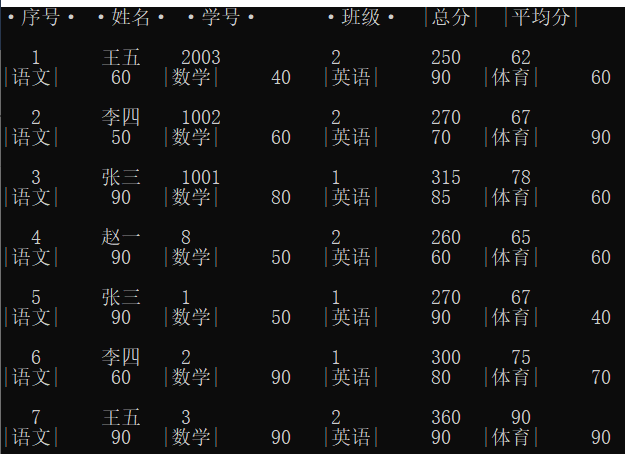
菜单界面



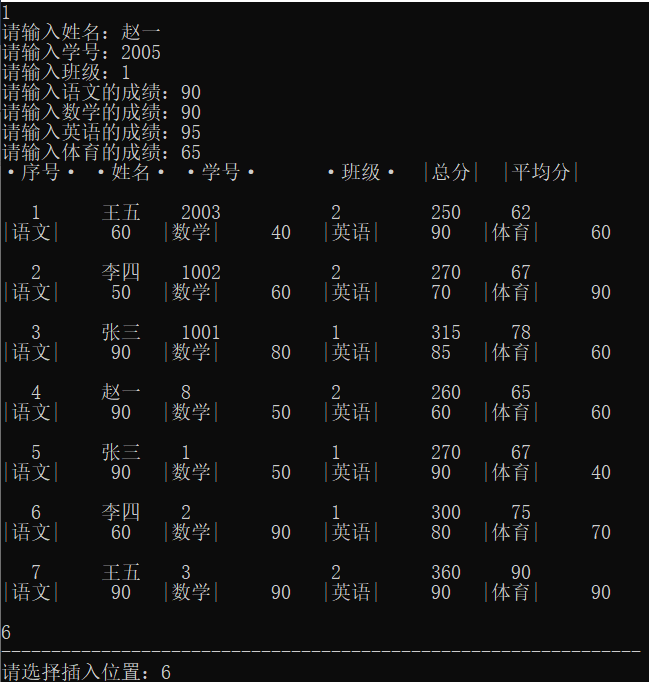
信息录入界面



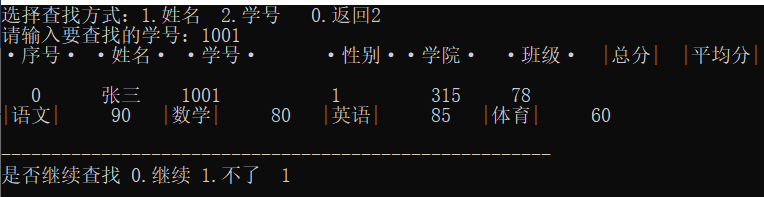
信息输出显示界面



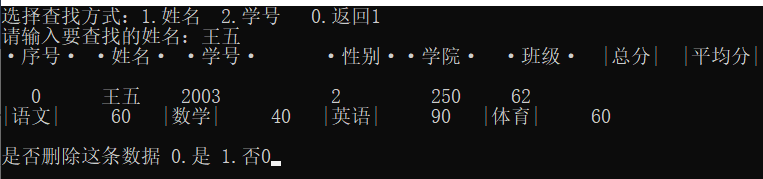
信息插入界面



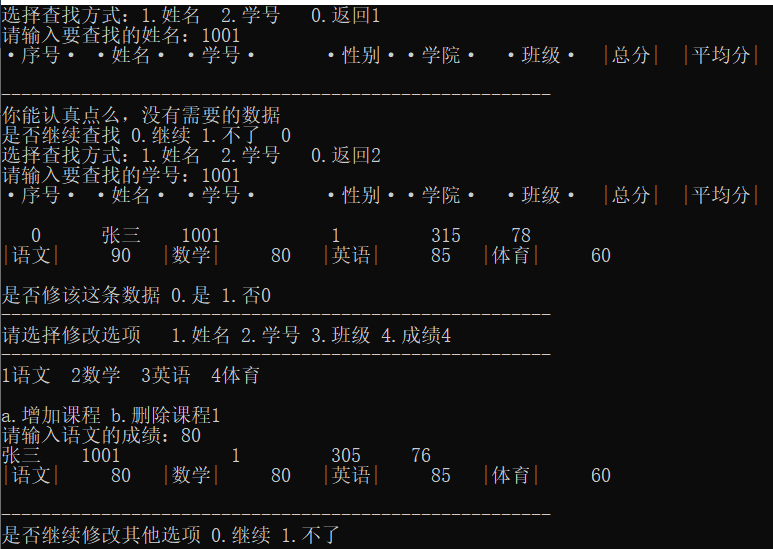
信息查找界面



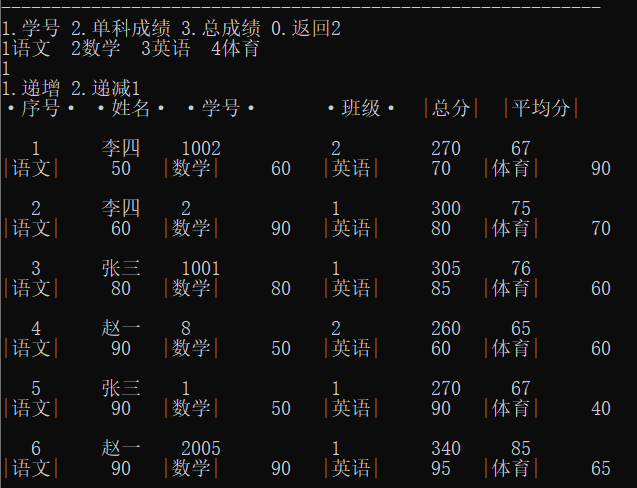
信息删除界面



信息修改界面



成绩排序界面





# 五．源程序清单

//2020.02

#include<iostream>

#include<string>

#include<fstream> //文件的输出输出流操作

#include<iomanip> //对齐输出

using namespace std;

const char\* filename="Student.txt"; //学生信息记录的文件名称

const char\* filename\_lesson="Lesson.txt"; //公共课程信息的记录名称

const int MAX=10; //最多课程数量

struct Lesson\_Node //课程的结构体设置

{

string L\_name; //课程名称

int grade; //课程成绩

};

class Lesson //课程设置类

{

private:

Lesson\_Node lesson[MAX]; //课程具体信息，最多10项

int n; //当前的课程数量

int Mount\_s; //成绩总和

int Averg\_s; //平均成绩

public:

Lesson() //课程初始化，读取文件中公共课程的信息设置,如果没有文件就建立一个文件

{

ifstream ifs(filename\_lesson);

if(!ifs)

{

string na[4]={"语文","数学","英语","体育"};

n=4;

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

{

lesson[i].L\_name=na[i];

lesson[i].grade=0;

}

SetA\_M();

Filein\_PublicL(); //如果打开失败，就写一个文件

}

else

{

ifs.close();

Fileout\_PbulicL(); //读取成功就读取当前文件中的课程设置

}

}

~Lesson() //析构

{

n=0;

Mount\_s=0;

Averg\_s=0;

}

Lesson(const Lesson&rhs) //复制构造函数

{

n=rhs.n;

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

{

lesson[i].L\_name=rhs.lesson[i].L\_name;

lesson[i].grade=rhs.lesson[i].grade;

SetA\_M();

}

}

Lesson& operator=(const Lesson&rhs)

{

n=rhs.n;

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

{

lesson[i].L\_name=rhs.lesson[i].L\_name;

lesson[i].grade=rhs.lesson[i].grade;

SetA\_M();

}

return \*this;

}

int ReturnMount\_s() //返回总分

{

return Mount\_s;

}

int ReturnAverg\_s() //返回平均分

{

return Averg\_s;

}

void Dislesson(int x=1) //输出当前的课程的信息

{

if(x==1) //由于需要，在没有输入参数的情况下默认是输出文件中的课程信息，在参数x不等于1则输出当前内存中修改后的课程程信息

Fileout\_PbulicL();

int i;

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

{

cout<<i+1<<lesson[i].L\_name<<" ";

}

cout<<endl;

}

void Addlesson(int i=1) //增加课程 参数1是公共课程的添加 2是私人课程的添加

{

char t='1';

string na;

while(t=='1')

{

if(n==MAX)

{

cout<<"以达到课程上限，无法继续添加"<<endl;

return ;

}

cout<<"输入要添加的课程名称：";

cin>>na;

lesson[n].L\_name=na;

if(i==1)

lesson[n].grade=0;

else

{

cout<<"输入成绩";

cin>>na;

lesson[n].grade=atoi(na.c\_str());

}

n++; //课程数量加1

cout<<"是否继续添加课程：（1.继续 0.不了）";

cin>>t;

}

if(i==1)

Filein\_PublicL();//将修改后的课程信息写入文件

}

void Dellesson(int i=1) //删除课程

{

int x;string tmp;char t='1';

while(t=='1')

{

cout<<"请输入要删除的课程编号";

cin>>tmp;x=atoi(tmp.c\_str());

if(x>0&&x<=n)

{

for(;x<n;x++)

{

lesson[x-1].L\_name=lesson[x].L\_name;

lesson[x-1].grade=lesson[x].grade;

}

n--;

if(i==1)

Dislesson(0); //显示当前内存中的课程信息，避免一直循环删除要一直读取写入文件，所以一次性将修改好的课程写入文件

}

else

{

cout<<"输入错误~~~，或已经不存在课程信息了"<<endl;

}

cout<<"是否继续删除课程信息（1.是的，0.不了） ";

cin>>t;

}

if(i==1)

Filein\_PublicL(); //写入文件

}

void SetA\_M() //计算平均分和总分

{

Mount\_s=0;

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

{

Mount\_s=Mount\_s+lesson[i].grade;

}

Averg\_s=Mount\_s/n;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 个人选修课程设置 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void SetPersonLesson()

{

string tmp;

char x='0';

int i;

Dislesson(2); cout<<endl; //输出当前学生信息中的课程信息

cout<<"a.增加课程 b.删除课程";

cin>>x;

switch (x)

{

case 'a':Addlesson(2);break; //增加单人课程

case 'b':Dellesson(2);break; //删除单人课程

default:

i=x-'0';

if(i>0&&i<=n)

{

cout<<"请输入"<<lesson[i-1].L\_name<<"的成绩：";

cin>>tmp;

lesson[i-1].grade=atoi(tmp.c\_str());//成绩从字符转整形

}

break;

}SetA\_M();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 公共课程设置部分 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Filein\_PublicL() //向文件中输入公共课程的设置

{

ofstream ofs(filename\_lesson);

if(!ofs)

{

cout<<"文件打开失败"<<endl;

exit(0);

}

ofs<<n<<'\t';

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

{

ofs<<lesson[i].L\_name<<'\t';

}

ofs.close();

}

void Fileout\_PbulicL() //从文件中读取公共课程设置信息，建立课程

{

int i=0;

ifstream ifs(filename\_lesson);

if(!ifs)

{

cout<<"文件打开失败"<<endl;

exit(0);

}

ifs>>n>>ws;

while(i<n&&!ifs.eof()) //判断是否读取到文件末尾了

{

ifs>>lesson[i].L\_name>>ws;

i++;

}

ifs.close();

}

void SetPbulicLesson() //设置当前的公共课程的信息

{

cout<<"当前公共课程信息："<<endl;//显示当前的课程信息便于用来

Dislesson();

cout<<"1.添加课程 2.删除课程 0.返回 ";

string tmp;char t;

cin>>t;

switch(t)

{

case '1':

Addlesson();

break;

case '2':

Dellesson();

break;

case '0':break;

}

}

friend istream& operator>>(istream& is,Lesson &rhs)//输入成绩

{

string tmp;

for(int i=0;i<rhs.n;i++)

{

cout<<"请输入"<<rhs.lesson[i].L\_name<<"的成绩：";

is>>tmp;

rhs.lesson[i].grade=atoi(tmp.c\_str());//成绩从字符转整形

}

return is;

}

friend ostream& operator<<(ostream& os,Lesson &rhs)//输出成绩

{

for(int i=0;i<rhs.n;i++)

{

cout<<"|"<<rhs.lesson[i].L\_name<<"|\t "<<rhs.lesson[i].grade<<"\t";

}

cout<<endl;

return os;

}

friend class student\_Node;

friend class student\_List;

};

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

Lesson L;//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*公共课程变量

class student\_Node //学生的信息类，存储学生的信息，并修改信息

{

private:

string Name; //姓名

long Number; //学号

string Class; //班级

Lesson Less; //课程

student\_Node \*next;

public:

student\_Node(string na="空",long num=0,string cla="空") //对成绩和姓名所以成员都进行初始化

:Less(L)

{

SetName(na);

SetNumber(num);

SetClass(cla);

next=NULL;

}

~student\_Node() //析构

{

Name="";Number=0;Class="";next=NULL;

}

student\_Node& operator =(const student\_Node&rhs)

{

Name=rhs.Name;

Number=rhs.Number;

Class=rhs.Class;

Less=rhs.Less;//这里又调用课程类的赋值函数

return \*this;

}

void SetName(string name) //设置名字

{

Name=name;

}

void SetNumber(long num ) //设置学号

{

Number=num;

}

void SetClass(string cla ) //设置班级

{

Class=cla;

}

void SetLesson() //单人课程设置

{

Less.SetPersonLesson();

}

friend istream& operator>>(istream&is,student\_Node &rhs)//输入数据

{

string tmp;

long num;

cout<<"请输入姓名："; //输入名字，到临时变量中，然后通过设置函数设置好成员中的名字，下面的都相同

is>>tmp;

rhs.SetName(tmp);

cout<<"请输入学号：";

is>>tmp;

num=atol(tmp.c\_str()); //字符型转化为长整形（解决了输入字符时会出现的bug）

rhs.SetNumber(num);

cout<<"请输入班级：";

is>>tmp;

rhs.SetClass(tmp);

is>>rhs.Less; //输入课程

return is;

}

friend ostream& operator<<(ostream&os,student\_Node &rhs)//输出数据

{

cout<<left<<setw(8)<<rhs.Name<<left<<setw(15)<<rhs.Number<<left<<setw(10)<<rhs.Class<<left<<setw(8)<<rhs.Less.ReturnMount\_s()<<left<<setw(8)<<rhs.Less.ReturnAverg\_s()<<endl;

cout<<rhs.Less<<endl; //输出课程

return os;

}

friend class student\_List; //友元类设置

};

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

class student\_List //学生的信息链表类，主要进行链表的操作，查找输出

{

private:

student\_Node \*head;

public:

student\_List() //学生链表的初始化，建立头结点,并从文件中读取信息建立链表

{

head=new student\_Node;

head->next=NULL;

ifstream ifs(filename);

if(!ifs) //如果文件不存在

{

Filewrite(); //建立文件

}

}

~student\_List()

{

student\_Node \*p=head->next,\*q;

while(p!=NULL)

{

q=p;

p=p->next;

delete q;

}

delete head;

}

void Fileread() //读取文件中的信息建立链表

{

ifstream ifs(filename);

if(!ifs)

{

cout<<"文件打开失败"<<endl;

exit(0);

}

student\_Node \*q=head;

while(ifs.peek()!=EOF) //判断是否读取到文件末尾了 采用这个peek函数而不采用ifs.eof()是防止文件为空时还进入循环

{

student\_Node \*p=new student\_Node;

ifs>>p->Name

>>ws

>>p->Number

>>ws

>>p->Class

>>ws;

ifs>>p->Less.n>>ws;

for(int i=0;i<p->Less.n;i++)

ifs>>p->Less.lesson[i].L\_name>>ws>>p->Less.lesson[i].grade>>ws;

p->Less.SetA\_M();

q->next=p;

p->next=NULL;

q=p;

}

}

void Filewrite() //通过文件的读写将链表中的内容写到文件中

{

student\_Node \*p=head->next;

ofstream ofs(filename);

if(!ofs)

{

cout<<"文件打开失败"<<endl;

exit(0);

}

while(p!=NULL)

{

ofs<<p->Name<<'\t'

<<p->Number<<'\t'<<p->Class

<<endl;

ofs<<p->Less.n;

for(int i=0;i<p->Less.n;i++)

ofs<<'\t'<<p->Less.lesson[i].L\_name<<'\t'<<p->Less.lesson[i].grade;

ofs<<endl;

p=p->next;

}

ofs.close();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 操作部分 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void Putin(int n=1) //输入,参数的含义是区别与是操作1的输入还好插入函数....默认是输入函数

{

char x='0';

int i=1; //计数

int num\_0,num\_1;//插入位置

student\_Node \*q;

if(n==1) //这是操作输入数据是需要的，如果插入数据则不需要

{

cout<<"是否清空原来数据 0.是的 1.不，添加进去 3.返回 ";

cin>>x;

if(x=='1')//添加进去

{

Fileread(); //读取文件形成链表

Putin(0); //参数的改变,就是插入函数的调用

return ;

}

}

if(x=='0')

{

while(x=='0')

{

cout<<i++<<endl;

student\_Node \*p=new student\_Node;

cin>>\*p;

if(n!=3)//节点的插法

{

p->next=head->next;

head->next=p;

}

else//节点的任意位置插入

{

Display(2);

cout<<"----------------------------------------------------------------"<<endl;

cout<<"请选择插入位置：";

cin>>x;

num\_0=x-'0';

q=head;

num\_1=0;//寻找位置

while (q->next!=NULL)

{

num\_1++;

if(num\_0==num\_1)

break;

q=q->next;

}

p->next=q->next;

q->next=p;

Display(2);

}

cout<<"请问是要继续么？（0.是 1.不是） ";

cin>>x;

cout<<"----------------------------------------------------------------"<<endl;

}

Filewrite();

}

}

void Display(int x=1) //显示输出cout<<left<<setw(40)<<"靠左输出"<<left<<setw(35)<<"靠左输出"<<endl;

{

if(x==1)

Fileread();

cout<<left<<setw(5)<<"·序号·"<<left<<setw(8)<<" ·姓名·"<<left<<setw(15)<<" ·学号·"<<left<<setw(10)<<"·班级·"<<left<<setw(8)<<"|总分|"<<left<<setw(8)<<"|平均分|"<<endl<<endl;

student\_Node \*p=head->next;

int i=1;

while(p!=NULL)

{

cout<<left<<setw(3)<<" "<<left<<setw(3)<<i++<<" "<<\*p;

p=p->next;

}

cin.get();

cin.get();

}

void Insert() //插入(添加)

{

Fileread(); //读取文件形成链表

Putin(3); //参数的改变,就是插入函数的调用

}

void Delete() //删除

{

Find(2);

Filewrite(); //写入文件

}

void Find(int i=1) //查找

{

Fileread();//读取文件形成链表

char x='0';

while(x=='0')

{

cout<<"选择查找方式：1.姓名 2.学号 0.返回";

cin>>x;

switch(x)

{

case '1':

Find\_01name(i);

break;

case '2':

Find\_02number(i);

break;

case '0':return;

default:

cout<<"选择错误！！！";

}

cout<<"是否继续查找 0.继续 1.不了 ";

cin>>x;

}

}

void Find\_01name(int x=1) //查找名字 参数 1.查找 2.删除 3.修改

{

int i=0;

string na;

char x\_0='1';

student\_Node \*p=head->next,\*q=head;

cout<<"请输入要查找的姓名：";

cin>>na;

cout<<left<<setw(5)<<"·序号·"<<left<<setw(8)<<" ·姓名·"<<left<<setw(15)<<" ·学号·"<<left<<setw(8)<<"·性别·"<<left<<setw(10)<<"·学院·"<<left<<setw(10)<<"·班级·"<<left<<setw(8)<<"|总分|"<<left<<setw(8)<<"|平均分|"<<endl<<endl;

while(p!=NULL)

{

if(p->Name==na)

{

cout<<left<<setw(3)<<" "<<left<<setw(3)<<i++<<" "<<\*p;

if(x==2)//删除数据

{

cout<<"是否删除这条数据 0.是 1.否";

cin>>x\_0;

if(x\_0=='0')

{

q->next=p->next;

delete p;

p=q;

}

}

if(x==3)//修改数据

{

cout<<"是否修该这条数据 0.是 1.否";

cin>>x\_0;

if(x\_0=='0')

{

Modify\_small(p);

}

}

}

q=p;

p=p->next;

}cout<<"-------------------------------------------------------"<<endl;

if(i==0)

cout<<"你能认真点么，没有需要的数据"<<endl;

}

void Find\_02number(int x=1)//学号查找

{

int i=0;

long num;

string tmp;

char x\_0='1';

student\_Node \*p=head->next,\*q=head;

cout<<"请输入要查找的学号：";

cin>>tmp;

num=atol(tmp.c\_str());

cout<<left<<setw(5)<<"·序号·"<<left<<setw(8)<<" ·姓名·"<<left<<setw(15)<<" ·学号·"<<left<<setw(8)<<"·性别·"<<left<<setw(10)<<"·学院·"<<left<<setw(10)<<"·班级·"<<left<<setw(8)<<"|总分|"<<left<<setw(8)<<"|平均分|"<<endl<<endl;

while(p!=NULL)

{

if(p->Number==num)

{

cout<<left<<setw(3)<<" "<<left<<setw(3)<<i++<<" "<<\*p;

if(x==2)//删除数据

{

cout<<"是否删除这条数据 0.是 1.否";

cin>>x\_0;

if(x\_0=='0')

{

q->next=p->next;

delete p;

p=q;

}

}

if(x==3)//修改数据

{

cout<<"是否修该这条数据 0.是 1.否";

cin>>x\_0;

if(x\_0=='0')

{

Modify\_small(p);

}

}

}

q=p;

p=p->next;

}

cout<<"-------------------------------------------------------"<<endl;

if(i==0)

cout<<"你能认真点么，没有需要的数据"<<endl;

}

void Modify() //修改 界面里加上一些划线增加美观度

{

Find(3);

Filewrite(); //写入文件

}

void Modify\_small(student\_Node \*p)

{

char a='0';

string tmp;

long num;

while(a=='0')

{

cout<<"-------------------------------------------------------"<<endl;

cout<<"请选择修改选项 1.姓名 2.学号 3.班级 4.成绩";

cin>>a;

cout<<"-------------------------------------------------------"<<endl;

switch(a)

{

case '1':

cout<<"输入名字：";

cin>>tmp;

p->SetName(tmp);

break;

case '2':

cout<<"输入学号：";

cin>>tmp;

num=atol(tmp.c\_str());

p->SetNumber(num);

break;

case '3':

cout<<"请输入班级：";

cin>>tmp;

p->SetClass(tmp);

break;

case '4':

p->SetLesson();//修改课程

break;

default:

cout<<"你可爱哦，输错啦，亲~"<<endl;

}

cout<<\*p;

cout<<"-------------------------------------------------------"<<endl;//美化视觉效果

cout<<"是否继续修改其他选项 0.继续 1.不了";

cin>>a;

}

}

void Sort() //排序

{

Fileread();

char t='0';

while (t=='0')

{

cout<<"------------------------------------------------------------"<<endl;

cout<<"1.学号 2.单科成绩 3.总成绩 0.返回";

cin>>t;

switch (t)

{

case '1':

Sort\_small(1);break;//学号排序

case '2':

Sort\_small(2);break;//单科成绩排序

case '3':

Sort\_small(3);break;//总成绩排序

default:

Filewrite();//

return ;

}

Display(0);

cout<<"是否继续排序 0.继续 1.不了,并保存结果";

cin>>t;

}

Filewrite(); //写入文件

}

void Sort\_small(int i=1)//排序函数分支，根据参数的不同进行不同选项的排序1.默认学号 2.单科成绩 3.总成绩

{

char t;int x;

if(i==2)//单科成绩排序时输出

{

L.Dislesson();//显示公共课程

cin>>t;

x=t-'0';//课程的下标

if(x>L.n||x<0)

{

cout<<"选择错误"<<endl;

return ;

}

}

cout<<"1.递增 2.递减";

student\_Node \*p1,\*p2;

student\_Node tmp;

int flag=1;

cin>>t;

while(1)//冒泡排序

{

flag=1;

p1=head->next;p2=p1->next;

while(p2!=NULL)

{

switch (t)

{

case '1':

if(i==3)//总分排序

if(p1->Less.Mount\_s>p2->Less.Mount\_s)

{

tmp=\*p1;

\*p1=\*p2;

\*p2=tmp;

flag=0;

}

if(i==2)//成绩排序

if(p1->Less.lesson[x-1].grade>p2->Less.lesson[x-1].grade)

{

tmp=\*p1;

\*p1=\*p2;

\*p2=tmp;

flag=0;

}

if(i==1)//学号排序

if(p1->Number>p2->Number)

{

tmp=\*p1;

\*p1=\*p2;

\*p2=tmp;

flag=0;

};break;

case '2':

if(i==3)//总分排序

if(p1->Less.Mount\_s<p2->Less.Mount\_s)

{

tmp=\*p1;

\*p1=\*p2;

\*p2=tmp;

flag=0;

}

if(i==2)//成绩排序

if(p1->Less.lesson[x-1].grade<p2->Less.lesson[x-1].grade)

{

tmp=\*p1;

\*p1=\*p2;

\*p2=tmp;

flag=0;

}

if(i==1)//学号排序

if(p1->Number<p2->Number)

{

tmp=\*p1;

\*p1=\*p2;

\*p2=tmp;

flag=0;

};break;

default:

break;

}

p1=p1->next;

p2=p2->next;

}

if(flag==1)

break;

}

}

void Cal() //计算各个学科总成绩和平均成绩 需要强大其功能

{

Fileread();

cout<<"-------------------------------------------------------------------"<<endl;

cout<<left<<setw(10)<<"科目"<<left<<setw(10)<<"总分"<<left<<setw(10)<<"平均分"<<left<<setw(10)<<"通过率"<<endl;//三个空格

for(int i=0;i<L.n;i++)

Cal\_small(L.lesson[i].L\_name);

cin.get();cin.get();

}

void Cal\_small(string na)

{

student\_Node \*p=head->next;

int num\_1=0,num\_n=0,num\_p=0;

while(p!=NULL)

{

for(int i=0;i<p->Less.n;++i)

{

if(p->Less.lesson[i].L\_name==na)

{

num\_n++;//总人数

num\_1=num\_1+p->Less.lesson[i].grade;

if(p->Less.lesson[i].grade>=60)

num\_p++;//通过人数

break;

}

}

p=p->next;

}

cout<<left<<setw(10)<<na<<left<<setw(10)<<num\_1<<left<<setw(10)<<num\_1/num\_n<<left<<setw(3)<<(float)num\_p/(float)num\_n\*100<<"%"<<endl;

}

float Grade\_fail(string name ,int grade) //计算成绩打败全校的百分比

{

int num\_1=0,num\_2=0;

float num;

student\_Node \*p=head->next;

while (p!=NULL)

{

for(int i=0;i<p->Less.n;i++)

{

if(name==p->Less.lesson[i].L\_name)//计算选这门课的人

{

num\_1++;

if(p->Less.lesson[i].grade<=grade)

num\_2++;//比你分数差的人人数

break;

}

}

p=p->next;

}

num=(float)num\_2/(float)num\_1\*100;//类型转换

return num;

}

};

const string Pass\_word="123456";//教师登陆密码

void Meue() //主菜单界面

{

while(1)

{

system("cls");

cout<<"\n\n\n\n\n";

cout<<" ▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁▁"<<endl;

cout<<" ▏ ▕"<<endl;

cout<<" ▏ 1.输入 ▕"<<endl;

cout<<" ▏ 2.输出 ▕"<<endl;

cout<<" ▏ 3.课程设置 ▕"<<endl;

cout<<" ▏ 4.添加 ▕"<<endl;//变成课任意插入位置的

cout<<" ▏ 5.查找 ▕"<<endl;

cout<<" ▏ 6.删除 ▕"<<endl;

cout<<" ▏ 7.修改 ▕"<<endl;

cout<<" ▏ 8.排序 ▕"<<endl;

cout<<" ▏ 9.计算 ▕"<<endl;

cout<<" ▏ 0.退出登陆 ▕"<<endl;

cout<<" ▏ ▕"<<endl;

cout<<" ▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"<<endl;

student\_List A;//建立链表,初始化

char t;

cin>>t;

system("cls");

switch (t)

{

case '1':

A.Putin();

break;

case '2':

A.Display();

break;

case '3':

L.SetPbulicLesson();

break;

case '4':

A.Insert();

break;

case '5':

A.Find();

break;

case '6':

A.Delete();

break;

case '7':

A.Modify();

break;

case '8':

A.Sort();

break;

case '9':

A.Cal();break;

case '0':return ;

default:

break;

}

}

}

void Welcome() //开始的登陆界面

{

system("cls");

cout<<"\n\n\n\n\n\n";

cout<<" ☆学生成绩排名系统☆"<<endl;

cout<<" ▇▇▇▇▇▇▇▇▇▇▇"<<endl;

cout<<" ★ ★"<<endl;

cout<<" ★ 1.登 录 ★"<<endl;

cout<<" ★ 0.退 出 ★"<<endl;

cout<<" ★ ★"<<endl;

cout<<" ▇▇▇▇▇▇▇▇▇▇▇";

cout << endl;

char x;

string tmp;

cin>>x;

switch(x)

{

case '0':exit(0);

case '1':

cout<<"\n\n 请输入密码：";

cin>>tmp;

if(tmp==Pass\_word) //判断是否通过密码

Meue();

break;

}

//\_sleep(2\*1000);//延时命令

}

int main()

{

while(1)

Welcome();

return 0;

}