沈阳航空航天大学

**高级程序设计实验报告**

课程设计名称：**面向过程程序设计综合应用**

课程设计题目：**学生信息管理系统**

学 院：计算机学院

专 业：软件工程

班 级： 1834010501

学 号： 183401050116

姓 名： 张鋆枻

成 绩：

指导教师： 许莉

完成时间：2019年5月31日

目录

[一、 实验目的 2](#_Toc11836000)

[二、 实验内容 2](#_Toc11836001)

[三、 实验要求 2](#_Toc11836002)

[3.1 实验总体要求 2](#_Toc11836003)

[3.2 实验功能要求 3](#_Toc11836004)

[四、 程序功能介绍 3](#_Toc11836005)

[五、 设计分析&流程 4](#_Toc11836006)

[表层 @Inerface 5](#_Toc11836007)

[**1、界面** 5](#_Toc11836008)

[**2、接底函数（表层界面对底层功能接口引擎函数）** 8](#_Toc11836009)

[**3、全局变量** 10](#_Toc11836010)

[里层-链表与操作 @List 11](#_Toc11836011)

[**1、链表的概念** 12](#_Toc11836012)

[**2、节点与节点概念** 12](#_Toc11836013)

[**3、创建新头结点 newhead()** 13](#_Toc11836014)

[**4、对节点操作** 14](#_Toc11836015)

[**5、对链表操作** 16](#_Toc11836016)

[六、试验结果 19](#_Toc11836017)

[1.搜索学生信息 19](#_Toc11836018)

[2.添加学生信息 20](#_Toc11836019)

[3.修改学生信息 21](#_Toc11836020)

[4.删除学生信息 22](#_Toc11836021)

[5.修改排序方式 23](#_Toc11836022)

[6.输入指令错误的应对： 24](#_Toc11836023)

[7.录入分数错误的应对： 25](#_Toc11836024)

[8.搜索无目标： 26](#_Toc11836025)

[七、设计感悟 27](#_Toc11836026)

[八、源代码 27](#_Toc11836027)

# 实验目的

本次《面向过程程序设计综合应用》实验，是基于“程序设计基础”和“高级程序设计”课程学习内容的重要实践环节，主要目的是通过综合性实验，培养和提高学生的独立分析问题、解决实际问题的能力和计算机语言编程能力。

# 实验内容

调研、选择一个典型学生学籍（成绩）信息管理案例，实现以班为单位的学生成绩信息管理。学生成绩的基本信息包括：学号、姓名、五门以上课程成绩、成绩总分、平均成绩等。

# 实验要求

### 3.1 实验总体要求

（1）阅读设计题目、任务内容，规划设计进度，并进行软件各相关功能模块的设计。

（2）在编译环境下，用结构化程序设计思想进行C语言程序设计、功能模块调试。

（3）进行实验设计报告编写与整理。

（4）实验结束时，要求进行成果演示（由老师验收相关程序运行成果并打分）；每人须上交实验报告（纸质、电子）。

### 3.2 实验功能要求

（1）输入/添加记录：能够输入、添加多条记录，程序中记录用链表进行存储。

（2）显示、查询记录：根据用户要求按所给记录关键字显示、查询一个或多个相关记录。

（3）修改记录：能对任意存在的记录进行修改，并能显示修改后的数据。

（4）写入文件：程序结束前，把存在的记录写入的磁盘文件中。

在正确、合理的软件功能规划的基础上，进行各功能模块的设计。切忌将多个功能模块混合一起进行调试。

# 程序功能介绍

程序主要功能为存储学生信息，学生信息包括学号，姓名，以及5种成绩。支持批量读入，多种排序，展示信息，删除，修改，存入文件，读取文件等一系列必要功能：

批量输入：允许用户批量录入学生信息，提升办公效率。

多种排序：程序提供基于学号或姓名的字典序排序，以及总分或单科成绩排序，可以在菜单切换。

信息展示：自动将所有录入信息清晰的展示出来。

搜索数据：支持通过学号或者姓名批量搜索目标数据，将所有符合的元素全部返回。

删除&修改数据：支持通过学号删除数据或修改数据。

数据同步：读取&存入文件功能应用实时同步，防止出现意外情况导致数据丢失。

在保证功能齐全的基础上做到操作界面清晰明了，操作方式简单友好。 同时在错误应对上采用了精密的算法，以应对用户的错误操作。

# 设计分析&流程

题目要求设计一款拥有输入，输出，存储，读入的链表数据组操作程序。

首先采用有顶向下的设计模式，先考虑出程序的表层功能。可以发现表层操作有大量重复的基础操作。将基础操作设计成一个一个基础函数，再用表层操作调用，增强程序的可扩充性，可升级性，增强代码的可读性。

将对链表的操作单独封装成一组函数，包括搜索、创建、添加、删除，对节点的操作单独封装成一组函数，包括输入，输出。

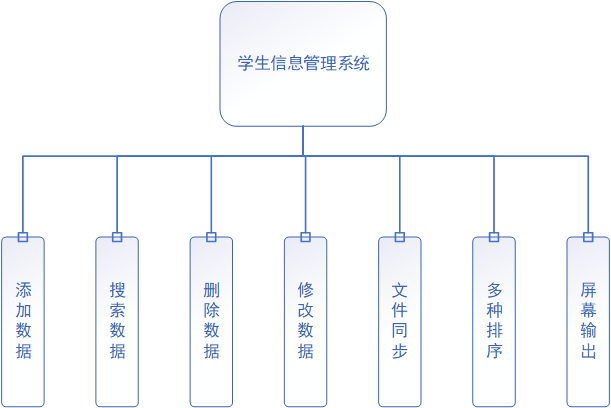


图5.1 程序功能模块图

### 表层 @Inerface

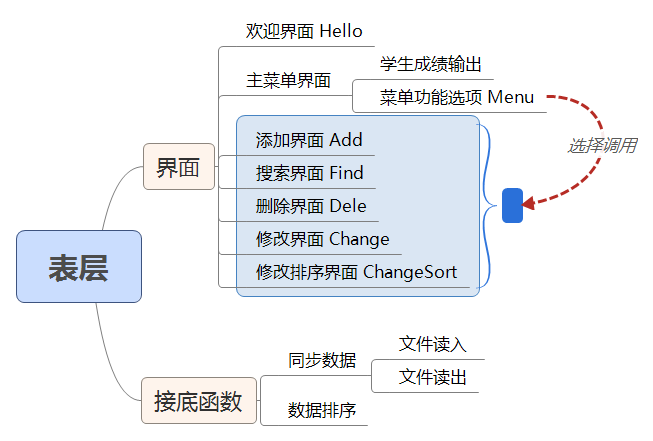


图5.2 表层结构图

**1、界面**

**1-1 界面板子**

* *板子的好处：*
  + 伟大的科学家曾经说过
  + 不能重复造轮子
  + 原因是反复做大量重复的东西不仅会**浪费时间**，而且**难以扩展和升级**。
  + 所以将重复的功能用**调用函数**代替，将重复的界面按规则构造，可以增加程序的可读性，可扩展性以及减少工程量
* *界面的通用性：*
  + 采用界面板子的方式，将界面中大量的**重复元素**进行**分类**。
  + 可以减少代码的重复性并**增强可扩展能力、可读性**。
* *板子结构：*
  + 预加载功能：
    - 对链表调出、重排序，刷新屏幕，同步数据等功能，在可视界面出现之前执行完成。
  + 可视界面：
    - 面向操作人员的提示&表达界面
  + 界面功能：
    - 对操作人员所进行的操作进行处理，通过调用底层功能实现应有反馈。

**1-2 欢迎界面 Hello()**

* 普通的欢迎界面，代码结构简单，只有显示功能。

**1-3 主菜单界面 Menu()**

* 负责显示菜单功能与程序信息。
* 功能：输出菜单功能与程序信息
* 输出程序的主页面功能信息：
* 需要调用排序变量SortMODE和条幅变量Text[]

**1-4 搜索元素界面 Find()**

* 根据输入的信息对学生进行搜索的功能界面
* 属于辅佐界面**调用不刷新界面**
* 功能：查询全部匹配值
  1. 读入用户的输入，并用其对数据进行匹配
  2. 将所有匹配成功的值输出
* 支持学号与姓名的**不完全匹配**搜索
* 需要调用底层功能：
  + finditems() 链表数据批量查询（不完全匹配搜索全部节点）
  + PrintList(forits)链表信息输出

**1-5 添加元素界面 Add()**

* 批量接受用户的输入，并反复添加学生信息
* **调用不刷新页面**
* 功能：输入新学生信息
  1. 反复输入学生信息并添加至数据中
  2. 直到输入为 ‘\*’字符
* 需要调用底层功能：
  + note::input\_scores() 节点信息输入
  + additem()向链表中插入节点

**1-6 删除元素界面 Dele()**

* 批量接受用户的输入，从数据中匹配元素，将匹配到的元素删除。
* **调用不刷新页面**
* 功能：批量删除元素
  1. 循环输入要删除元素的学号信息，直到输入 ‘\*’字符
  2. 匹配成功则将匹配成功的元素删除，否者反馈不存在
* 需要调用底层功能：
  + finditem()链表寻找单一节点（单节点不完全匹配）
  + deleitem()链表删除节点

**1-7 修改元素界面 Change()**

* 接受用户的一个输入，将匹配的元素输出，并执行修改操作。
* **调用不刷新界面**
* 功能：修改指定元素
  + 输入修改目标的学号
  + 返回匹配目标，若无匹配则返回**未找到**
  + 对目标数据对应修改
* 需要调用底层功能
  + finditem()链表寻找单一节点（单节点不完全匹配）
  + note::output()单一节点信息输出
  + **支持万用字符‘-’**

**1-8 修改排序界面 Changesort()**

* 根据用户的选择，改变面板输出的排序方式
* 调用会跳转到新界面
* 功能：修改排序方式
  + 根据用户的选择，**修改排序方式SortMODE值**，以改变排序方式
  + 提供 总分排序、学号排序、单科排序
* 排序的修改**只作用于当前面板**，不影响存档和下一次启动
* 调用全局变量SortMODE

**1-9 错误应对输入**

* 为防止用户误输入导致程序错误，采用**错误应对输入**
* 功能：读取一行字符串，只截取有用的部分，若无效则继续输入
* 嵌入所有指令输入的位置

**2、接底函数（表层界面对底层功能接口引擎函数）**

**2-1 排行榜输出 PrintStus()**

* 按顺序输出当前主链表中所有元素信息
* 调用函数会清屏
* 功能：输出主链表中的全部元素信息
  + 调用条幅函数TextLine()输出条幅
  + 调用底层链表输出函数PrintList(note\*)输出主链表

**2-2 输出条幅 TextLine()**

* 输出条幅
* 参数接口
  + bool TestMode = **false**,是否为测试模式输出（文件输出）
* 功能：输出条幅
  + 按照宏定义域宽输出文字“学号”、“姓名”，再调用全局科目参数 Text[]将其内容依次输出
  + 构成输出数据的头顶条幅
  + 测试模式将**取消域宽**

**2-3 目标链表排序 Sort()**

* 对目标链表进行排序
* 参数接口
  + note \*head,排序目标的头节点
  + int sm = 0,排序方式，根据sm的值选择[0-6]的排序方式
* 功能：对指定的**头节点所引出的链表**进行**指定方式的排序**
* 排序方式函数：
  + 使用自定义排序cmp
  + 0 - 总分降序
  + 1 - 学号升序
  + [2-6] - 单科降序
* 需要调用底层功能：
  + sort\_list() 对链表排序

**2-5 数据读入 structin()**

* 从数据文件中读入元素信息
* 功能：从文件中读入元素信息
  + 对数据文件进行**穷尽读入**，将所有读入元素输入运行内存的数据
* 需要调用底层功能：
  + note::input\_scores() 节点信息输入
  + additem()向链表中插入节点

**2-6 数据输出 structout()**

* 向数据文件输出全部元素信息
* 功能：向数据文件输出全部元素信息
  + 将当前数据内的所有元素**有序覆盖**入数据文件
* 需要调用底层功能：
  + note::output() 单一节点数据输出
  + lishead 主链表头结点
  + 功能直接对接底层链表，加快同步速度

**3、全局变量**

**3-1 Text[]**

* 科目名称变量
* char **Text[20] [20]** = {"总分","程序","高数","数逻","离散","思政"};

**3-2 SortMODE**

* 排序方式变量，值分别对应各种排序方式
* **值可以被改变 [0-6]**
* int **SortMODE** = 0;
  1. **listhead**
* 主链表头部 note **\*listhead**；
  1. **forits**
* 缓存链表头部 note **\*forits**；

### 里层-链表与操作 @List

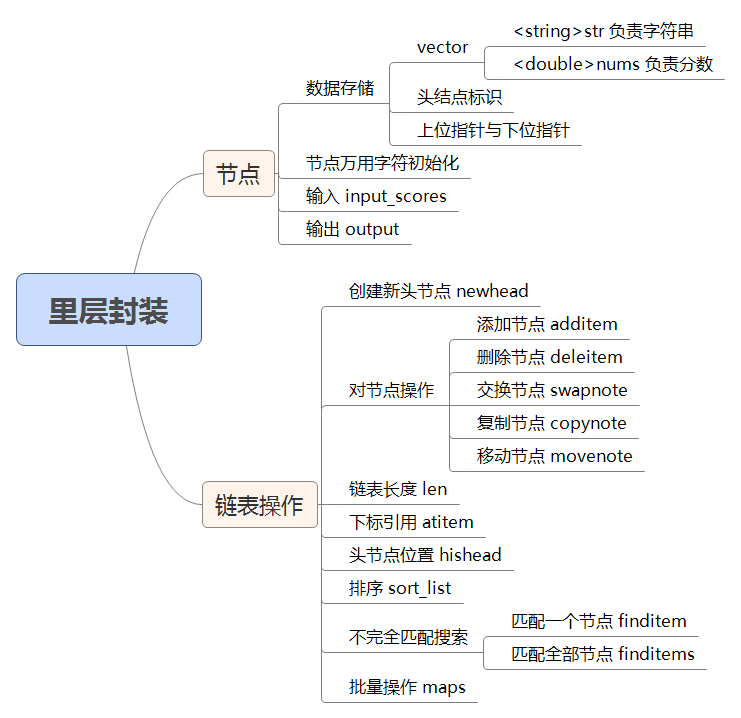


图5.3 里层封装

**1、链表的概念**

* 链表采用**有头双向循环链表**
* 头节点只用于引用链表，无法进行任何单独操作

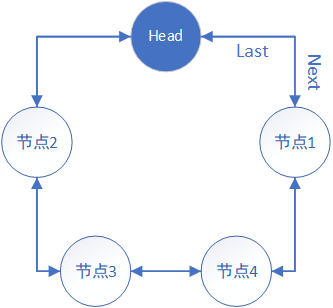
****

图5.4 链表结构

**2、节点与节点概念**

**2-1 节点内容**

* 节点采用字符串与数字的分开存储
* 定义双**vector**容器
  + strs
    - strs[0] 学号
    - strs[1] 姓名
  + nums
    - nums[0] 总分
    - nums[1:6] 单科分数
* 是否为头结点标识
* 上位指针与下位指针

**2-2 节点初始化**

* 初始化节点无分数，学号与姓名为**万用字符 “-"**

**2-3 节点输入与输出**

*（1）输入分数 input\_scores()*

* 从目标源读入全部成绩并计算出总分
* 使用方法：item.**input\_scores()**;

*（2）输出信息 output()*

* 向目标源按 *学号 姓名 分数* 输出节点信息
* 使用方法：item.**output()**;

**2-4 万用字符与节点比较**

* “-”为万用字符，与任意字符串进行**==**运算均为**true**

**3、创建新头结点 newhead()**

* 赋予节点空间，并转化为头节点
* 使用方法：note \*head = **newhead()**;

**4、对节点操作**

**4-1 添加节点 additem()**

* 在目标位置的前面插入元素
* 使用方法：**additem(p,item)**；或者**additem(p,\*item)**;
* 如果目标位置为**头节点**则相当于在**链表尾部插入**

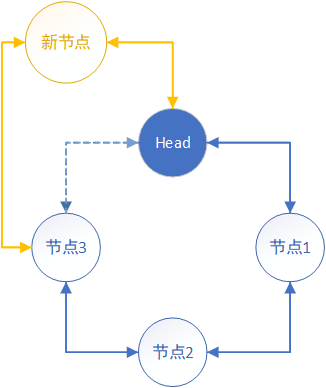


图5.5 添加节点

**4-2 删除节点 deleitem()**

* 删除目标位置的节点从链表中
* 使用方法：**deleitem(p)**;
* **不能删除头结点**

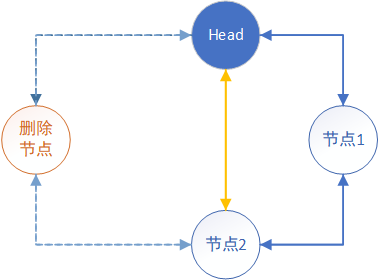
****

图5.6 删除节点

**4-3 交换节点 swapnote()**

* 交换两个位置的节点
* 使用方法： **swapnote(p,q)**;
* **不能交换头结点**

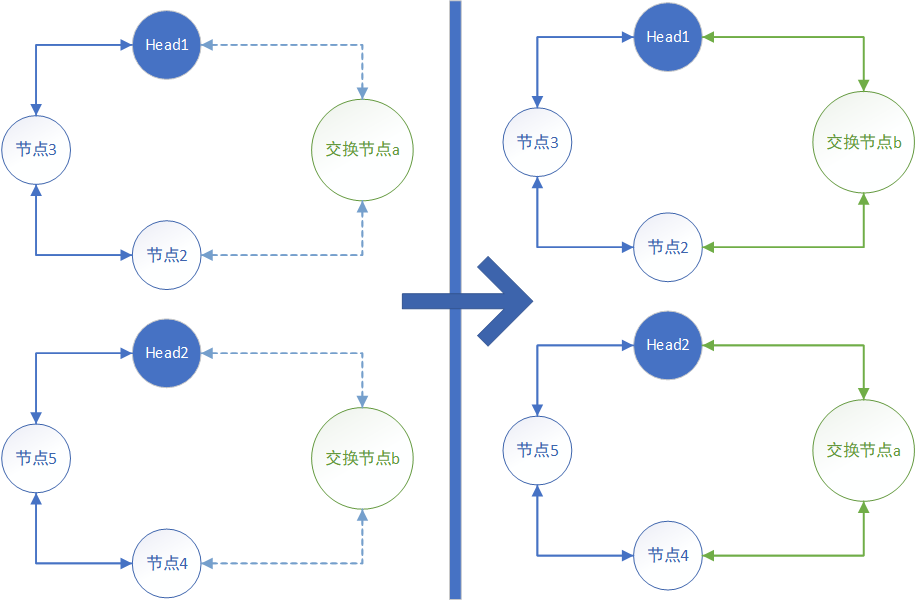


图5.7 交换节点位置

**4-4 复制节点 copynote()**

* 返回目标位置节点的复制
* 使用方法：note item = **copynote(p)**;
* **不能复制头节点**

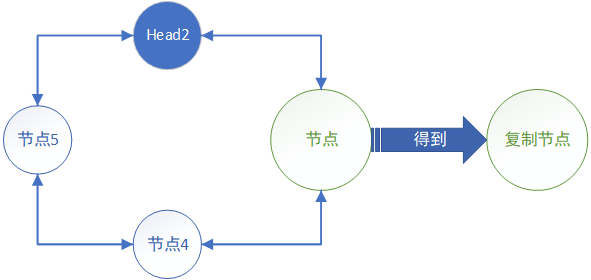


图5.8 复制节点

**4-5 移动节点 movenote()**

* 将目标位置2节点移动到目标位置1的前面
* 使用方法**movenote(p1,p2)**;
* **不能移动头节点**

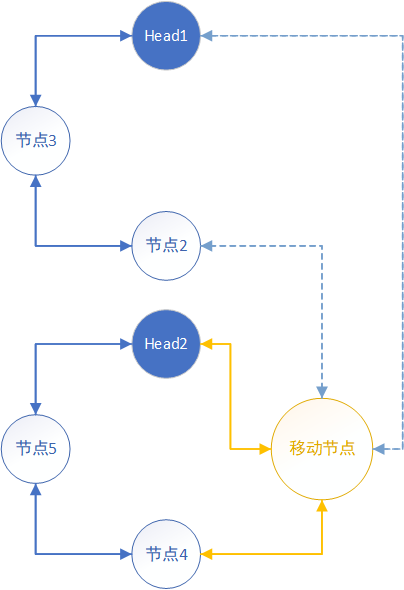


图5.9 移动节点

**5、对链表操作**

**5-1 链表长度 len()**

* 返回链表长度
* 使用方法: **len(head)**;
* **必须以头结点为参数**

**5-2 下标引用 atitem()**

* 返回正向第n个元素的地址
* 使用方法：note \*p = **atitem(head,n)**;
* **必须以头结点为参数**

**5-3 头位置 hishead()**

* 返回位置元素的头节点
* 使用方法：note head = **hishead(p)**;

**5-4 排序 sort\_list()**

* 将目标链表按照**自定义比较**方式排序
* 使用方法 **sort\_list(head,cmp)**;
* **必须引用头节点**

**5-5 不完全匹配搜索**

*（1）匹配一个节点 finditem()*

* 按照**不完全匹配搜索**方式，将正向遍历到的第一个匹配的节点位置返回
* 使用方法 note \*p = **finditem(head,item)**;

*（2）匹配全部节点 finditems()*

* 按照**不完全匹配搜索**方式，将所有匹配成功的节点存入辅助链表
* 引用参数
  + note **\*head** 搜索的起点
  + note **\*forits** 辅助链表
  + note **item** 匹配元素
  + bool **copy** = true 是否为复制（是否移除原链表中的节点）
  + bool **mult** = true 在辅助链表中的元素是否可重复
* 使用方法：**finditems(head,forist,item)**;

*（3）不完全匹配*

* 应用万能字符“-”，将匹配元素中**无关变量**设置为“-”
* 在比较元素时只考虑**主要变量**即可

**5-6 批量操作 maps()**

* 对**目标链表**进行**指定操作**，直到**指定目标**达成
* 引用参数
  + note **\*position** 接入位置
  + void **(\*mapfuntion)(note \*)** 批量操作函数
  + bool **Zheng** = true 是否为正向
  + bool **( \*iffunction)(note \*,note \* )** = is*once*cycle 终止条件
* 使用方法：**maps(head,printitem)**;

# 六、试验结果

### 1.搜索学生信息

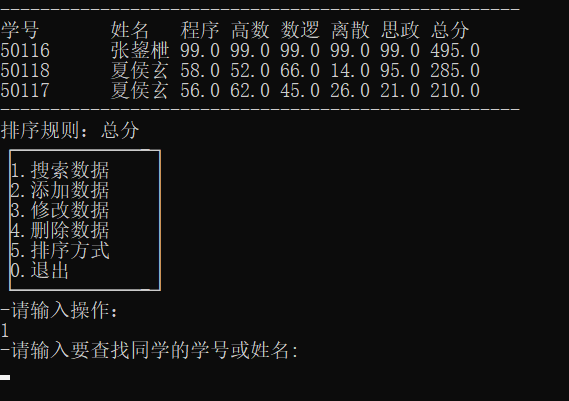


图6.1 搜索调用

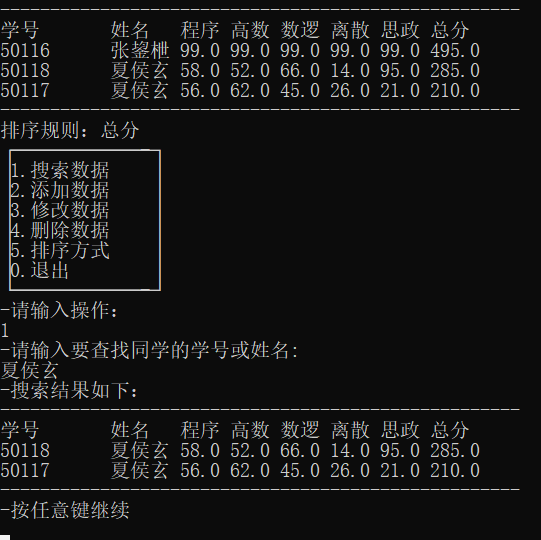


图6.2 搜索结果

### 2.添加学生信息

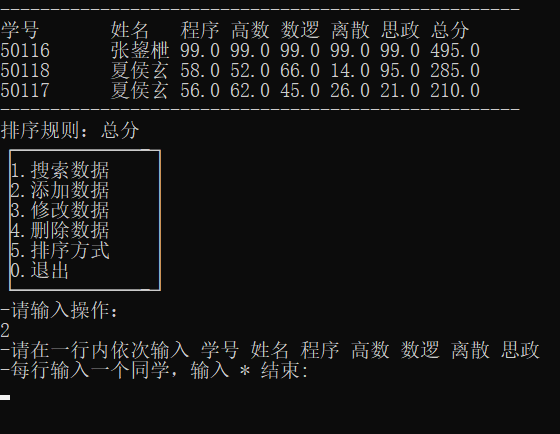


图6.3 添加调用

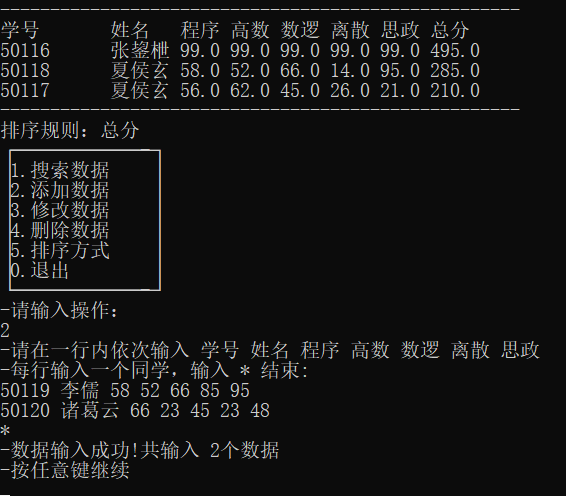


图6.4 添加结果

### 3.修改学生信息

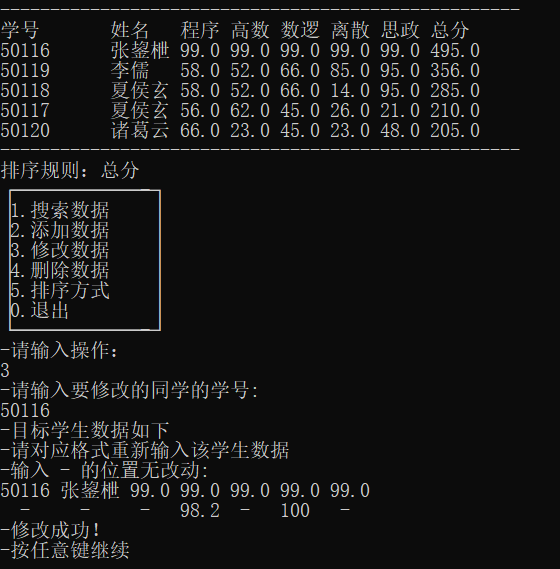


图6.5 修改数据

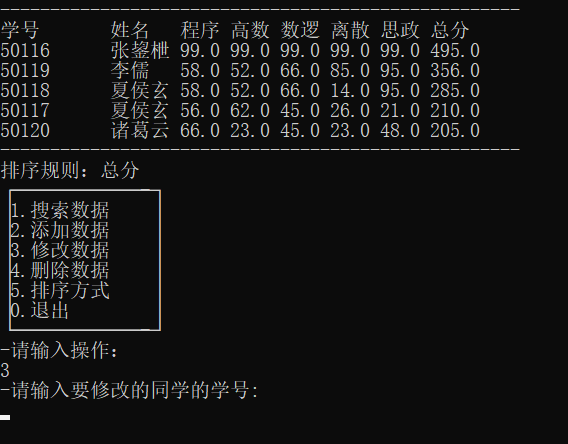


图6.6 修改结果

### 4.删除学生信息

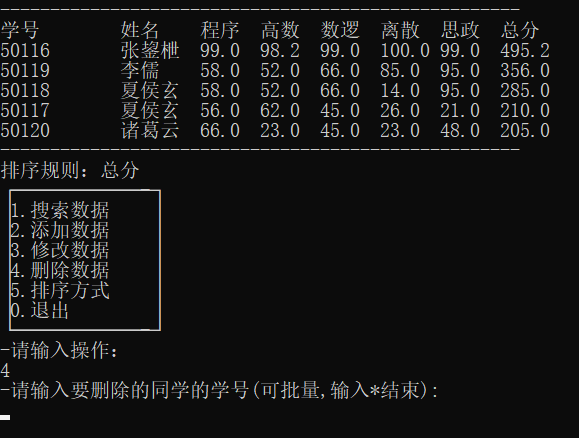


图6.7 删除调用

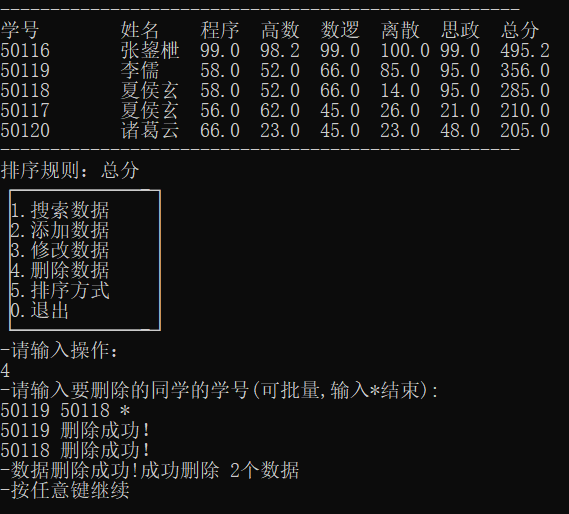


图6.8 删除结果

### 5.修改排序方式

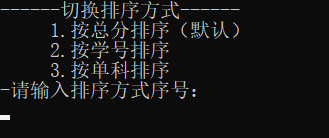


图6.9 修改排序方式

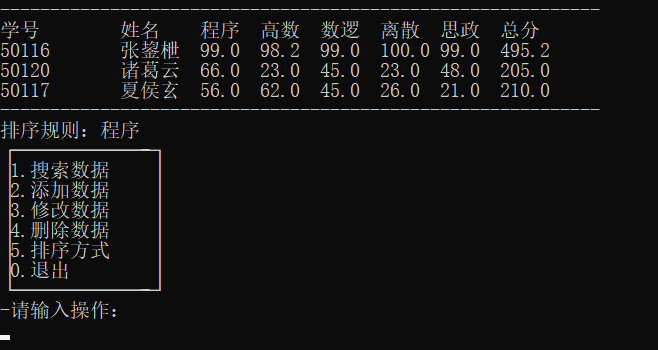


图6.10 修改排序结果

### 6.输入指令错误的应对：

#### 对于各页面输入的指令行，只读取第一个字符，并将其转化为指令数字。如果转化失败或者无目标指令则重新输入新指令

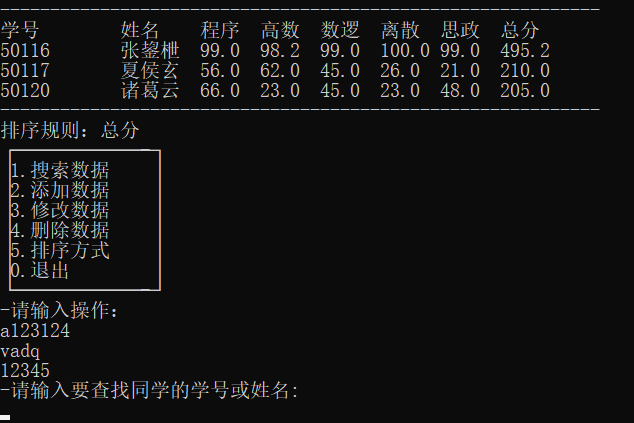


图6.11 输入指令的错误应对

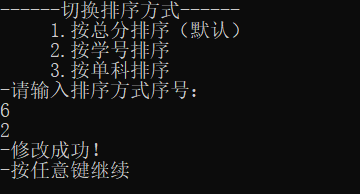


图6.12 修改排序的错误应对

### 7.录入分数错误的应对：

#### 如果在添加或修改时录入分数时输入了非数字字符，则会在输入错误的成绩位置抛出-1错误数字替代

#### 

图6.13 成绩录入错误

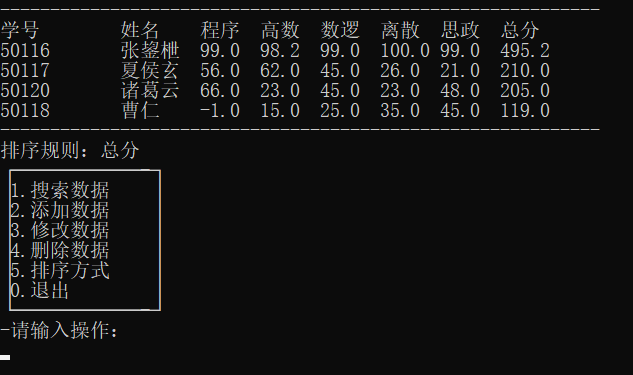


图6.14 成绩录入的错误应对

### 8.搜索无目标：

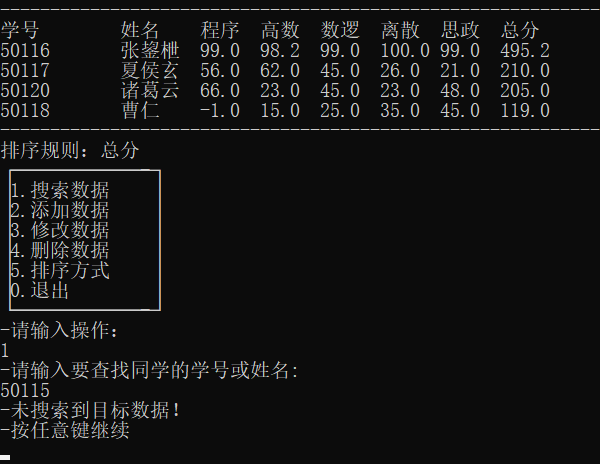


图6.15 搜索无目标

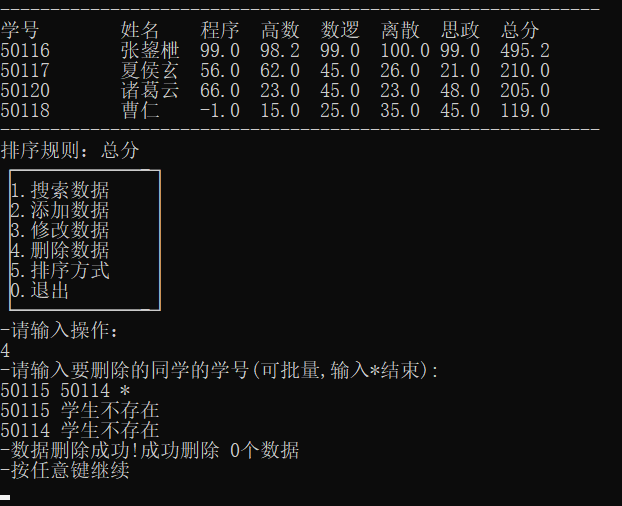


图6.16 删除无目标

# 七、设计感悟

此次课设题目让我充分体会到了自顶向下的设计模式，以及面向对象的编程方式和模块化设计的魅力。

充分的锻炼对实际任务的应对。了解了数据在数据中的传输，存储方式，同时学会了C语言的高级语法与技巧。

# 八、源代码

#include <bits/stdc++.h>  
#include <windows.h>  
#define fr(i,k) for(int i=0;i<k;i++)  
#define frr(i,k,n) for(int i=k;i<n;i++)  
  
#pragma execution\_character\_set("utf-8")  
  
///参数调整  
#define bottonnum 5 ///按键数  
 ///域宽  
#define numW 12 ///学号  
#define nameW 8 ///姓名  
#define scoreW 6 ///分数  
  
using namespace std;  
///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///  
///--- @Programfunction ---辅助操作函数-----------------------------------///  
  
 ///程序调试&&运行错误退出  
void exit\_sl(int a=0){  
 system("pause>nul");  
 exit(a);  
}  
  
 ///string == 重载  
bool operator == (string a,string b)  
{  
 if(a.length()!=b.length())  
 return 0;  
 fr(i,a.length())  
 if(a[i]!=b[i])  
 return 0;  
 return 1;  
}  
  
bool operator < (string a,string b)  
{  
 int la = a.length();  
 int lb = b.length();  
 if(la!=lb)  
 return la<lb;  
  
 fr(i,la){  
 if(a[i]!=b[i])  
 return a[i]<b[i];  
 }  
  
 return 0;  
}  
  
  
 ///string != 重载  
bool operator != (string a,string b)  
{  
 if(a.length()!=b.length())  
 return 1;  
 fr(i,a.length())  
 if(a[i]!=b[i])  
 return 1;  
 return 0;  
}  
  
  
 ///顿字输出语句  
void print(string s,char end='\n')  
{  
 int l = s.length();  
 int time = 230/l;  
 fr(i,l){  
 cout<<s[i];  
 Sleep(time);  
 }  
 cout<<end;  
}  
  
 ///模仿jv string + 运算  
string operator + (string a,string b)  
{  
 return a.append(b);  
}  
  
 ///暂停  
void pause(bool printstr = true)  
{  
 if(printstr)  
 print("-按任意键继续");  
 system("pause>nul");  
}  
  
 ///清屏  
void clearscreen(int Sleeptime = 0)  
{  
 Sleep(Sleeptime);  
 system("cls");  
}  
  
  
 ///string -> double 转换  
double todouble(string s)  
{  
 int l = s.length();  
 fr(i,l)  
 if(!(('0'<=s[i]&&s[i]<='9')||s[i]=='.'))  
 return -1;  
  
 stringstream ss;  
 ss<<s<<endl;  
 double b;  
 ss>>b;  
 return b;  
}  
  
///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///  
///--- @List ---链表和链表操作函数---------------------------------------///  
  
  
class note ///链表节点单元  
{  
public:  
 ///结构体内文本类数据，- 为任意数据  
 ///0 num 1 name  
 vector<string>strs;  
 ///结构体内数字类数据，- 为任意数据  
 ///0 all [1:6) score  
 vector<double>nums;  
  
 bool ishead; ///是否为头节点  
 int len; ///链表长度  
  
 note \*next;  
 note \*last;  
  
 note(){ishead=false;len=0;strs.push\_back("-");strs.push\_back("-");}  
 ~note(){};  
  
 void output(bool TestMode = false,ostream &out = cout) ///输出结构体  
 {  
 if(!TestMode){  
 out<<setw(numW)<<left<<strs[0];  
 out<<setw(nameW)<<left<<strs[1];  
 frr(i,1,6)  
 out<<setw(scoreW)<<left<<nums[i];  
 out<<setw(scoreW+1)<<left<<nums[0];  
 cout<<endl;  
 }  
 else{  
 out<<strs[0]<<' ';  
 out<<strs[1]<<' ';  
 frr(i,1,6)  
 out<<nums[i]<<' ';  
 cout<<endl;  
 }  
 }  
  
 void input\_scores(int l = 5,istream &in = cin) ///输入分数  
 {  
 string inp;  
 nums.push\_back(0);  
 fr(i,l){  
 in>>inp;  
 nums.push\_back(todouble(inp));  
 nums[0]+=todouble(inp);  
 }  
 }  
};  
  
note \*newhead();  
void movenote(note \*a,note \*b);  
void swapnote(note \*a,note \*b);  
note copynote(note \*a);  
note \*atitem(note \*head,int n);  
void deleitem(note \*position);  
note \*hishead(note\* p);  
  
 ///创建新头  
note \*newhead()  
{  
 note \*nhead = new note;  
 nhead->next = nhead; ///空头 自成环  
 nhead->last = nhead;  
 nhead->ishead = true;  
 nhead->len = 0;  
 return nhead;  
}  
  
 ///重载note::strs == 比较  
bool operator == (note a,note b)  
{  
 int la = a.strs.size(),lb = b.strs.size();  
 if(la!=lb)  
 return false;  
 fr(i,la){  
 if(a.strs[i]!=b.strs[i]&&a.strs[i]!="-"&&b.strs[i]!="-") /// - 语句表示万能语句，可以与所有语句相等  
 return false;  
 }  
  
 return true;  
}  
  
 ///将a移动到b的前面  
void movenote(note \*b,note \*a)  
{  
 if(a->ishead==true){  
 cout<<"movenote:错误！移动的节点是头结点"<<endl;  
 exit\_sl(0);  
 }  
  
 if(a==b) return;  
 a->last->next = a->next;  
 a->next->last = a->last;  
  
// cout<<a->num[0] <<' '<<b->num[0] <<endl;  
  
 b->last->next = a;  
 a->last = b->last;  
 b->last = a;  
 a->next = b;  
}  
  
 ///交换a与b的位置  
void swapnote(note \*a,note \*b)  
{  
 if(a->ishead==true||b->ishead==true){  
 cout<<"swapnote:错误！交换的节点是头结点"<<endl;  
 exit\_sl(0);  
 }  
  
 a->next->last = b;  
 a->last->next = b;  
 b->last->next = a;  
 b->next->last = a;  
 note \*temp = b->next;  
 b->next = a->next;  
 a->next = temp;  
 temp = b->last;  
 b->last = a->last;  
 a->last = temp;  
}  
  
 ///返回note a 的复制元素  
note copynote(note \*a)  
{  
 if(a->ishead==true){  
 cout<<"swapnote:错误！复制的节点是头结点"<<endl;  
 exit\_sl(0);  
 }  
  
 note item;  
  
 item.strs=a->strs;  
 item.nums=a->nums;  
 return item;  
}  
  
 ///在位置前面添加数据  
void additem(note \*position,note iteml)  
{  
 ///循环链表，head->last 是尾部  
 note \*item = new note(iteml);  
  
 item->next=position;  
 position->last->next = item;  
 item->last = position->last;  
 position->last = item;  
 //hishead(position)->len++;  
 item->len=item->last->len+1;  
}  
  
 ///在位置前面添加数据  
void additem(note \*position,note \*item)  
{  
 ///循环链表，head->last 是尾部  
 item->next=position;  
 position->last->next = item;  
 item->last = position->last;  
 position->last = item;  
 //hishead(position)->len++;  
 item->len=item->last->len+1;  
}  
  
 ///删除目标位置的数据  
void deleitem(note \*position)  
{  
 if(position->ishead==true){  
 cout<<"deleitem:错误！删除头节点不合法"<<endl;  
 exit\_sl(0);  
 }  
  
  
 position->last->next = position->next;  
 position->next->last = position->last;  
 delete position;  
}  
  
 ///返回链表长度  
int len(note \*head)  
{  
 if(head->ishead==false){  
 cout<<"len:错误！引用的链表接入点不是头结点"<<endl;  
 exit\_sl(0);  
 }  
  
 if(head->last==head)  
 return 0;  
  
 return head->last->len;  
}  
  
  
  
///----搜索函数--------///  
  
 ///查找第n位学生（循环），返回其地址  
note \*atitem(note \*head,int n)  
{  
 if(head->ishead==false){  
 cout<<"atitem:错误！引用的链表接入点不是头结点"<<endl;  
 exit\_sl(0);  
 }  
  
 int num=0;  
 note \*p = head->next; ///head不算位  
 while(true){ ///0123456排开  
 if(n==num)  
 return p;  
 if(p->ishead==false)  
 num++;  
 p = p->next;  
 }  
  
}  
  
 ///搜索并返回p的头节点  
note \*hishead(note\* p)  
{  
 note \*h = p;  
 while(h->ishead==false){  
 h = h->next;  
 }  
 return h;  
}  
  
 ///返回一个与值匹配的节点  
note \*finditem(note \*head,note item)  
{  
 note \*p = head->next;  
 while(p!=head){  
 if(\*p==item)  
 return p;  
 p=p->next;  
 }  
 return NULL;  
}  
  
 ///用缓存链表载入一群与值匹配的节点,是否为复制,是否允许重复？  
void \*finditems(note \*head,note \*forits,note item,bool copy = true,bool mult = false)  
{  
 note \*p = head->next;  
 while(p!=head){  
 if(\*p==item){  
 if(copy){  
 if(mult==true)  
 additem(forits,copynote(p));  
 else if(mult==false&&finditem(forits,\*p)==NULL)  
 additem(forits,copynote(p));  
  
 p=p->next;  
 }  
 else{  
 note \*frps=p;  
 p=p->next;  
 movenote(forits,frps);  
 }  
 }  
 else{  
 p=p->next;  
 }  
 }  
}  
  
///----排序----///  
  
 ///对链表使用cmp排序方式（插入排序法 On2）  
void sort\_list(note \*head,bool (\*cmpfunction)(note\*,note\*))  
{  
 if(head->ishead==false){  
 cout<<"sort\_list:错误！搜索引用的链表接入点不是头结点"<<endl;  
 exit\_sl(0);  
 }  
  
 note \*p = head->next;  
 while(p!=head){  
 note \*j = p->next;  
 while(j!=head){  
 if(!cmpfunction(p,j)){  
 p->nums.swap(j->nums);  
 p->strs.swap(j->strs);  
 }  
 j=j->next;  
 }  
 p=p->next;  
 }  
}  
  
  
  
bool is\_once\_cycle(note \*begin\_position,note \*p);  
void printitem(note \*p);  
  
///----批量操作----///  
  
 ///对链表执行mapf操作，终止条件iffun：起始位置,操作函数，终止条件，正向？  
void maps( note \*position, void (\*mapfunction)(note\*), bool Zheng = true, bool (\*iffunction)(note\*,note\*) = is\_once\_cycle)  
{  
 if(Zheng){ ///从position位置开始对整个链表操作  
 note \*p = position;  
 while(true){  
 if(p->ishead==false)  
 mapfunction(p);  
 p = p->next;  
 if(iffunction(position,p))  
 break;  
 }  
 }  
 else{  
 note \*p = position;  
 while(true){  
 if(p->ishead==false)  
 mapfunction(p);  
 p = p->last;  
 if(iffunction(position,p))  
 break;  
 }  
 }  
}  
  
 ///打印链表  
void PrintList(note \*head)  
{  
 maps(head,printitem);  
 cout<<"------------------------------------------------------------"<<endl;  
}  
  
///--自定义的循环终止条件函数--  
bool is\_once\_cycle(note \*begin\_position,note \*p)///是否为一圈？（只循环一圈）  
{  
 if(p == begin\_position)  
 return true;  
 return false;  
}  
  
bool is\_one\_left(note \*bp,note \*p)  
{  
 if(bp->next = bp)  
 return true;  
 return false;  
}  
  
///--自定义可被批量的操作函数（要求无返回值，只有参数note\*）  
void printitem(note \*p) ///打印节点  
{  
 p->output();  
}  
  
///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///  
///--- @Inerface ---界面&程序表面--------------------------------------------------------------------///  
  
  
char Text[20][20] = {"总分","程序","高数","数逻","离散","思政"};  
note \*listhead,\*forits; ///主链表头，缓存链表头  
  
 ///输出横幅  
void TextLine(bool TestMode = false)  
{  
 if(len(listhead)){  
 if(!TestMode){  
 cout<<"------------------------------------------------------------"<<endl;  
 cout<<setw(numW)<<left<<"学号";  
 cout<<setw(nameW)<<left<<"姓名";  
 frr(i,1,6)  
 cout<<setw(scoreW)<<left<<Text[i];  
 cout<<setw(scoreW+1)<<left<<Text[0];  
 cout<<endl;  
 }else{  
 cout<<"学号"<<' ';  
 cout<<"姓名"<<' ';  
 frr(i,1,6)  
 cout<<Text[i]<<' ';  
 cout<<endl;  
  
 }  
 }  
 else{  
 cout<<"\*\*\*数据为空\*\*\*"<<endl;  
 }  
}  
  
  
 ///排序规则  
int SortMODE = 0;  
  
  
 ///主菜单  
void Menu()  
{  
 cout<<"排序规则：";  
 if(SortMODE == 0) cout<<"总分";  
 else if(SortMODE == 1) cout<<"学号";  
 else cout<<Text[SortMODE-1];  
 cout<<endl;  
  
  
 cout<<"┌─────────────-┐"<<endl;  
 cout<<"│1.搜索数据 │"<<endl;  
 cout<<"│2.添加数据 │"<<endl;  
 cout<<"│3.修改数据 │"<<endl;  
 cout<<"│4.删除数据 │"<<endl;  
 cout<<"│5.排序方式 │"<<endl;  
 cout<<"│0.退出 │"<<endl;  
 cout<<"└─────────────-┘"<<endl;  
}  
  
 ///欢迎界面  
void Hello()  
{  
 clearscreen();  
 cout<<"-----------------------"<<endl;  
 cout<<" 欢迎来到班级管理系统"<<endl;  
 cout<<"-----------------------"<<endl;  
 print("-按任意键进入");  
 pause(false);  
 cout<<endl;  
}  
  
 ///搜索界面  
void Find()  
{  
 note item1,item2;  
 print("-请输入要查找同学的学号或姓名:");  
 string inp;  
 cin>>inp;  
 item1.strs[0]=inp;  
 item2.strs[1]=inp;  
  
 maps(forits,deleitem,true,is\_one\_left);///清空缓存  
 forits = newhead();  
  
 finditems(listhead,forits,item1);  
 finditems(listhead,forits,item2);  
  
 if(!len(forits)){  
 print("-未搜索到目标数据！");  
 }  
 else{  
 print("-搜索结果如下：");  
 TextLine();  
 PrintList(forits);  
 }  
 pause();  
  
 maps(forits,deleitem,true,is\_one\_left);///清空缓存  
 forits = newhead();  
}  
  
 ///删除界面  
void Dele()  
{  
 string inp;  
 print("-请输入要删除的同学的学号(可批量,输入\*结束):");  
  
 int sum=0;  
 while(true){  
  
 note item;  
 cin>>inp;  
 if(inp == "\*")  
 break;  
 if(inp == "-")  
 inp = "&";  
 item.strs[0]=inp;  
 note \*p = finditem(listhead,item);  
  
 cout<<inp<<' ';  
 if(p == NULL){  
 print("学生不存在");  
 }  
 else{  
 deleitem(p);  
 print("删除成功！");  
 sum++;  
 }  
  
 }  
 print("-数据删除成功!成功删除",' ');  
 cout<<sum;  
 print("个数据");  
 pause();  
}  
  
 ///添加界面  
void Add()  
{  
 print("-请在一行内依次输入 学号 姓名 程序 高数 数逻 离散 思政");  
 print("-每行输入一个同学，输入 \* 结束:");  
 int sum = 0;  
 while(true){  
 note item;  
 string inp;  
 cin>>inp;  
 if(inp == "\*")  
 break;  
 item.strs[0]=inp;  
 cin>>item.strs[1];  
 item.input\_scores();  
  
 additem(listhead,item);///向链表里添加节点  
 sum++;  
 }  
  
 print("-数据输入成功!共输入",' ');  
 cout<<sum;  
 print("个数据");  
 pause();  
}  
  
 ///修改界面  
void Change()  
{  
 string inp;  
 print("-请输入要修改的同学的学号:");  
 cin>>inp;  
 note item;item.strs[0]=inp;  
 note \*p = finditem(listhead,item); ///在主链中寻找节点  
 if(p!=NULL){  
 print("-目标学生数据如下");  
 print("-请对应格式重新输入该学生数据");  
 print("-输入 - 的位置无改动:");  
 p->output(true);  
  
 cin>>inp;  
 if(inp!="-")  
 p->strs[0]=inp;  
 cin>>inp;  
 if(inp!="-")  
 p->strs[1]=inp;  
  
 frr(i,1,5){  
 cin>>inp;  
 if(inp!="-")  
 p->nums[i]=todouble(inp);  
 }  
 print("-修改成功！");  
 }  
 else{  
 print("-未找到目标数据！");  
 }  
 pause();  
}  
  
 ///输出学生排行榜  
void PrintStus()  
{  
 clearscreen();  
 TextLine();  
 PrintList(listhead);  
}  
  
 ///改变排序界面  
void Changesort()  
{  
 int bottonnum\_1=3,bottonnum\_2=5;  
 clearscreen();  
 cout<<"------切换排序方式------"<<endl;  
 cout<<" 1.按总分排序（默认）"<<endl;  
 cout<<" 2.按学号排序"<<endl;  
 cout<<" 3.按单科排序"<<endl;  
 print("-请输入排序方式序号：");  
  
 int b;  
 string inp;  
 do{  
 getline(cin,inp);  
 b=inp[0]-'0';  
 if(0<=b&&b<=bottonnum\_1)  
 break;  
 }while(true);  
  
 if(b==1||b==2){  
 SortMODE = b-1;  
 }  
 else{  
 cout<<"-----------------------------------"<<endl;  
 cout<<"1.程序 2.高数 3.数逻 4.离散 5.思政"<<endl;  
 print("-请输入排序引用的单科序号：");  
 do{  
 getline(cin,inp);  
 b=inp[0]-'0';  
 if(0<=b&&b<=bottonnum\_2)  
 break;  
 }while(true);  
  
 SortMODE = b+1;  
 }  
 print("-修改成功！");  
 pause();  
  
}  
  
  
///--自定义数据排序方式--//  
  
bool cmp\_all(note \*a,note \*b) //总分  
{  
 return a->nums[0]>b->nums[0];  
}  
  
bool cmp\_stunum(note \*a,note \*b) //学号  
{  
 return a->strs[0] < a->strs[0];  
}  
  
bool cmp\_sco1(note \*a,note \*b) //分数1  
{  
 return a->nums[1]>b->nums[1];  
}  
  
bool cmp\_sco2(note \*a,note \*b) //2  
{  
 return a->nums[2]>b->nums[2];  
}  
  
bool cmp\_sco3(note \*a,note \*b) //3  
{  
 return a->nums[3]>b->nums[3];  
}  
  
bool cmp\_sco4(note \*a,note \*b) //4  
{  
 return a->nums[4]>b->nums[4];  
}  
  
bool cmp\_sco5(note \*a,note \*b) //5  
{  
 return a->nums[5]>b->nums[5];  
}  
  
 ///对序列使用Sm方式排序  
void Sort(note \*head,int Sm=0)  
{  
 bool (\*cmpFuntion)(note\*,note\*);  
  
 switch(Sm){  
 case 0:cmpFuntion = cmp\_all;break;  
 case 1:cmpFuntion = cmp\_stunum;break;  
  
 case 2:cmpFuntion = cmp\_sco1;break;  
 case 3:cmpFuntion = cmp\_sco2;break;  
 case 4:cmpFuntion = cmp\_sco3;break;  
 case 5:cmpFuntion = cmp\_sco4;break;  
 case 6:cmpFuntion = cmp\_sco5;break;  
 }  
  
 sort\_list(head,cmpFuntion);  
}  
  
  
  
///--数据同步--//  
#define datefile "date.dat"  
  
 ///从数据中同步数据  
void structin()  
{  
 ifstream datf(datefile,ios::in);  
  
 string inp;  
 while(datf>>inp){ ///定向datf流输入  
  
 note item;  
 item.strs[0]=inp;  
 datf>>item.strs[1];  
 item.input\_scores(5,datf);  
  
 additem(listhead,item);  
 }  
 datf.close();  
}  
  
 ///向数据更新数据  
void structout()  
{  
 ofstream datf(datefile,ios::out);  
  
 note \*p = listhead->next;  
 while(p!=listhead){  
 p->output(true,datf);  
 p=p->next;  
 }  
 datf.close();  
}

int main()  
{  
 cout<<setiosflags(ios::fixed)<<setprecision(1);  
  
  
 listhead = newhead();  
 forits = newhead();  
  
 Hello(); ///@Hello 加载欢迎界面  
 structin(); ///@structin 载入数据  
 string inp;  
 while(true){  
 Sort(listhead,SortMODE); ///@Sort 重排序  
 PrintStus(); ///@PrintStus 输出学生信息  
 Menu(); ///@Menu 加载菜单界面  
 print("-请输入操作：");  
 int b;  
 do{  
 getline(cin,inp);  
 b=inp[0]-'0';  
 if(0<=b&&b<=bottonnum)  
 break;  
 }while(true);  
 switch(b){  
 case 0:  
 exit(0); ///@exit 退出  
 case 1:  
 Find();break; ///@Find 查找  
 case 2:  
 Add();break; ///@Add 添加  
 case 3:  
 Change();break; ///@Change 修改  
 case 4:  
 Dele();break; ///@Dele 删除  
 case 5:  
 Changesort();break; ///@Changesort 修改排序方式  
 }  
 structout(); ///@structout 同步数据  
 }  
 return 0;  
}