浙江工业大学

C++程序设计课程设计
课设报告

2023/2024(2)

实验题目 学生成绩管理系统

学生姓名 _____

学生学号 _____

学生班级 _____

任课教师 _____

提交日期 2024 年 5 月 15 日

计算机科学与技术学院

学生成绩管理系统 实验报告

一、 实验内容及要求

学生成绩管理系统（SPMS：Student Performance Management System）用于管理学生的基本信息以及其成绩情况，要求完成的主要的功能为学生基本信息管理。可以完成录入学生信息、修改删除学生信息、统计学生成绩、计算平均分、按成绩排序、查询学生信息等工作。要求使用学习过的 C++程序设计的知识完成学生成绩管理系统的设计与实现。

二、 运行环境

学生成绩管理系统（SPMS）在 CLion 2023.3.2 平台下开发，操作系统：Windows 11。

硬件环境：

处理器：13th Gen Intel(R) Core(TM) i9-13900HX 2.20 GHz

内存：16.0 GB

系统类型：64 位操作系统

三、 实验课题分析（主要的模块功能、流程图）

3.1 学生成绩管理系统（SPMS）的主要功能

学生成绩管理系统（SPMS）**主要功能为：**学生成绩管理、账号（教师和管理员账号）管理。教师账号可以完成自己班级下学生信息录入、查询（支持模糊查询）、排序（学号、单科成绩、平均成绩、总分等）、统计、修改学生信息、删除学生信息等工作；管理员账号可以在教师账号的基础上完成所有学生信息管理、添加删除教师账号、重置账号密码、修改账号状态等工作。详细的系统功能结构为图 1 所示。

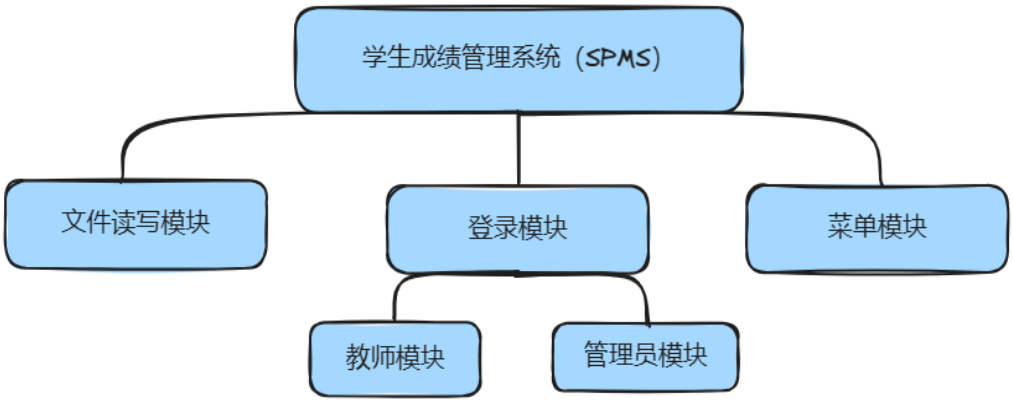


图 1 系统结构图

系统各模块的功能具体描述为：

1、 文件读写模块

文件读写模块分为文件环境全局配置、文件初始化读入、文件写入。在系统的开始，配置文件的路径和读写的文件类型。通过文件环境全局配置，可以方便更换系统的文件读写目标路径和目标文件的类型。文件初始化读入，学生信息修改删除等操作均需调用文件读写模块完成数据修改和更新。

2、 菜单模块

菜单模块在系统成功启动后进行加载，将主菜单、功能菜单等进行封装。一方面可以使代码清晰易懂；另一方面，也可以实现系统相关代码的封装和保密。

3、 登录模块

启动系统后，会从文件读入所有的信息，包括学生信息、教师和管理员账号。进入登录模块后，用户选择登录身份（教师、管理员），输入用户名和密码，用户名和密码均正确，即为登录成功，进行相应的功能模块。每个账号有五次输入机会，如果五次机会后密码仍输入错误，该账号将会被锁定，需要由管理员进行解除。

退出登录的时候则在退出系统前，保存当前系统的所有状态，包括学生信息和当前所有账号，然后安全退出系统。

4、 教师模块

教师在输入正确的账号和密码后，获取到对应的功能菜单。在本系统中设定每个教师管理一个班级，拥有对该班级学生信息管理的权限。在该模块下，教师可以进行以下操作：添加学生信息、查询学生信息、排序学生成绩、统计学生成绩、修改学生信息、删除学生信息。在查询学生信息功能中，教师可以根据学生的学号，班级进行精确查找，也可以根据姓名进行模糊查询；在排序学生成绩功能中，教师可以根据学生的学号、单科成绩、平均成绩、总分进行排序。

5、 管理员模块

管理员在输入正确的账号和密码后，获取到对应的功能菜单。管理员拥有对所有学生信息管理的权限，在教师功能的基础上可以对所有学生的信息进行添加、查询、排序、统计、修改、删除等操作。在统计功能中，管理员可以获取到所有班级学生成绩的基本情况，包括班级人数、单科成绩平均分、总分平均分等。

此外，管理员还可以对教师账号进行管理，包括添加教师账号、删除教师账号、重置账号密码、修改账号状态等。

3.2 系统分析及设计

系统涉及对象有五个类：菜单类、学生类、管理员类、学生链表类、管理员链表类。

用文本文件进行数据的保存，需要保存的数据主要包括学生信息（学生姓名、学生学号、学生性别、学生班级、学生成绩（高数、程 C、离散、大物）、总分、平均分）、账号（包

括教师、管理员的账号密码、用户类型、账号状态、管理的班级）。设置数据操作类，实现所有的文本操作相关的功能。

3.3 系统的实现

（1）类的编写

系统工程名为：SPMS。包含了 Menu 类（菜单类），Student 类（学生类），Manager 类（管理员类）三个基本类，以及相对应的链表类 Stulist（学生链表类）及 Manlist（管理员链表类）。给管理员设置 userType，用以区分教师和管理员；设置 status，用以表示账号状态（活跃、锁定）；设置 classmanage，用以表示教师所管理的班级。

具体类结构声明如下：

- **Menu 类**

```
class Menu {
public:
    static void welcome();

    static void teacherMenu();

    static void managerMenu();

    static void bye();

    static void howFind();

    static void howCount();

    static void howSort();

    static void countScore();

private:
    static void clearScreen() {
        system("cls");
    }
};
```

- **Student 类**

```
class Student {
private:
    string name;
    string gender;
    string id;
    string className;
    int scores[4];
```

```
    int totalScore;
    double averageScore;
public:
    Student *next;

    // 构造函数
    Student(string namev = "", string genderv = "", string idv = "", string cla
ssNamev = "")
            : name(namev), gender(genderv), id(idv), className(classNamev), tot
alScore(0), averageScore(0.0) {
        for (int i = 0; i < 4; ++i) {
            scores[i] = 0;
        }
    }

    string getName() const { return name; }

    string getGender() const { return gender; }

    string getId() const { return id; }

    string getClassName() const { return className; }

    const int *getScores() const { return scores; }

    int getTotalScore() const { return totalScore; }

    double getAverageScore() const { return averageScore; }

    void setName(const string &namev) { name = namev; }

    void setGender(const string &genderv) { gender = genderv; }

    void setId(const string &idv) { id = idv; }

    void setClassName(const string &classNamev) { className = classNamev; }

    void setScores(const int (&scoresv)[4]) { memcpy(scores, scoresv, sizeof(sc
ores)); }

    // 计算总分
    void calculateTotalScore();

    // 计算个人平均分
```

```
    void calculateAverageScore();

    // 初始化学生成绩
    void readScores(int (&scores)[4]);
};
```

- **Manager 类**

```
class Manager {
public:
    Manager *next;

    Manager(string usernamev = "", string passwordv = "", int userTypev = 1, int statusv = 1, string classmanagev = "")
            : username(usernamev),
              password(passwordv),
              userType(userTypev),
              status(statusv), classmanage(classmanagev) {};//构造函数

    string getUsername() { return username; }//获取账号

    string getPassword() { return password; }//获取密码

    int getUserType() { return userType; }//获取用户类型

    int getStatus() { return status; }//获取账号状态

    string getClassManage() { return classmanage; }//获取权限

    void setStatus(int s) { status = s; }//修改账号状态

    void setPassword(string passwordv) { password = passwordv; }

private:
    string username;//账号
    string password;//密码
    int userType; //用户类型(0:初始化 1:老师;2:管理员)
    int status; //账号状态(1:活跃;2:锁定)
    string classmanage; //管理的班级(管理员为0;老师为对应的班级)
};
```

（2）链表的使用

系统实现采用文件的输入输出流对文本数据进行读取与写入，但是由于学生信息、管理员账号都是一个数据的集合，于是对数据的存储组织使用了单向链表。

- **Stulist 类**

```
class Stulist {
```

```
private:
    Student *head;
    int size;
public:
    Stulist() {
        head = new Student;
        head->next = nullptr;
        size = 0;
    }

    Stulist(const Stulist &other) {
        head = new Student;
        head->next = nullptr;
        size = 0;

        Student *p = other.head->next;
        Student *tail = head;
        while (p) {
            Student *newNode = new Student(*p);

            tail->next = newNode;
            tail = newNode;

            p = p->next;
        }
    }

    // 获取表头
    Student *getHead() { return head; }

    // 添加学生
    void addStudent(string &namev, string &genderv, string &idv, string &cl
assNamev, int (&scoresv)[4]);

    // 老师添加学生
    void addStudentByTeacher(string classmanagev);

    // 查看学生是否已经存在
    Student *checkIfExist(string idv);

    // 写入文件
    void write();

    // 根据学号查找学生
```

```
        void findStudentByID(string classmanagev);

        // 根据姓名查找学生(模糊查找)
        void findStudentsByName(string classmanagev);

        // 根据班级查找学生
        void findStudentsByClass(string classmanagev);

        // 根据学号升序排序
        void sortStudentsByID(string classmanagev);

        // 根据单科成绩降序排序
        void sortStudentsBySubjectScore(string classmanagev);

        // 根据总分降序排序
        void sortStudentsByTotalScore(string classmanagev);

        // 根据平均分降序排序
        void sortStudentsByAverageScore(string classmanagev);

        // 统计学生信息
        void countStudentsScore(string classmanagev);

        // 查找同一个班的学生
        Student *findTheSameClassStudents(std::string classmanagev);

        // 修改学生信息
        void updateStudentByID(string classmanagev);

        // 删除学生信息
        void deleteStudentByID(string classmanagev);

        // 展示所有学生
        void showAllStudents();

    };
```

在运用时，令当前 Student 的 next 结点指向新的 Student 结点，即结点的指针 next 保存新的 Student 结点的地址（如下图 2 所示），以此类推，所有 Student 信息就通过链表的形式串联起来了。

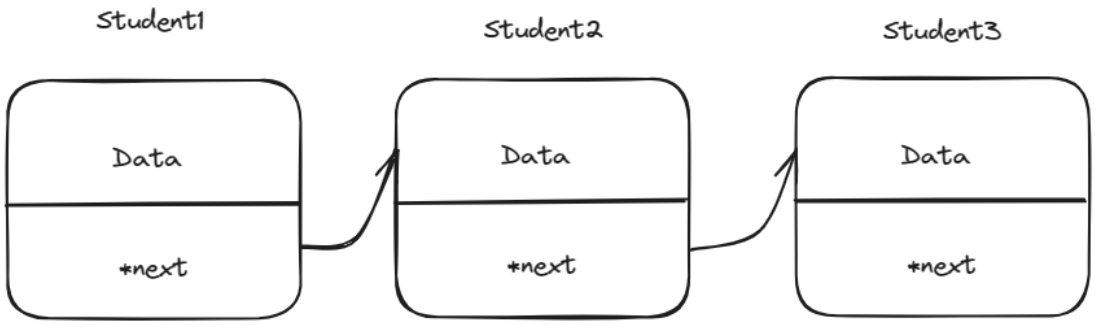


图 2 学生链表的建立

管理员类（Manager）的链表设置与学生的类似，也是通过定义类的指针变量，通过指向下一个类的地址将信息串联起来。即在 Manager 的基础上定义 Manlist，结构声明如下：

- **Manlist 类**

```
class Manlist {
private:
    Manager *head;
    int size;
public:
    Manlist() {
        head = new Manager;
        head->next = nullptr;
        size = 0;
    }

    // 添加账号
    void addManager(string &usernamev, string &passwordv, int &userTypev, int &statusv, string &classmanagev);

    // 登录
    Manager *login();

    // 管理员添加教师账号
    void addTeacherByManager();

    // 检查登录
    Manager *checkIfLogin(string usernamev, string passwordv);

    // 检查账号是否存在
    Manager *checkIfExist(string usernamev);

    // 查找账号是否存在
    Manager *findManagerByUsername(const string &username);
```

```
    // 管理员添加学生
    void addStudentByManager(Stulist &stulistv);

    // 管理员根据学号查询学生
    void managerFindStudentByID(Stulist &stulistv);

    // 管理员根据姓名查询学生
    void managerFindStudentsByName(Stulist &stulistv);

    // 管理员根据班级查询学生
    void managerFindStudentsByClass(Stulist &stulistv);

    // 管理员根据学号升序排序学生
    void managerSortStudentsByID(Stulist &stulistv);

    // 管理员根据单科成绩降序排序学生
    void managerSortStudentsBySubjectScore(Stulist &stulistv);

    // 管理员根据总分降序排序学生
    void managerSortStudentsByTotalScore(Stulist &stulistv);

    // 管理员根据个人平均分降序排序学生
    void managerSortStudentsByAverageScore(Stulist &stulistv);

    // 管理员根据班级统计学生成绩
    void managerCountStudentsScore(Stulist &stulistv);

    // 管理员修改学生信息
    void managerUpdateStudentByID(Stulist &stulistv);

    // 管理员删除学生信息
    void managerDeleteStudentByID(Stulist &stulistv);

    // 删除账号
    void delManager();

    // 重置账号密码
    void resetPassword();

    // 写入文件
    void write();

    // 锁定账号
```

```
        bool lockAccount(const string &username);

        // 修改账号状态
        void setAccount();

        // 获取所有账号
        void showAllManagers();
    };
```

学生成绩管理系统的信息的管理就具体表现为链表的操作。拿学生信息来说，学生信息的查找、修改、添加、删除与链表的查找、修改、添加、删除对应。

- **学生信息的查找**

教师在查询学生信息时，有三种不同的模式，见图 3：

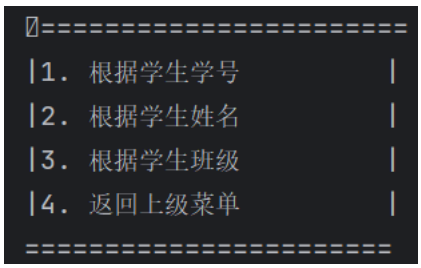


图 3 学生信息查找界面

这三种情况都需要对链表中的所有数据进行顺序的搜索。 例如按照学生学号的查找，定义一个 Student 类指针变量 p 并对其初始化。

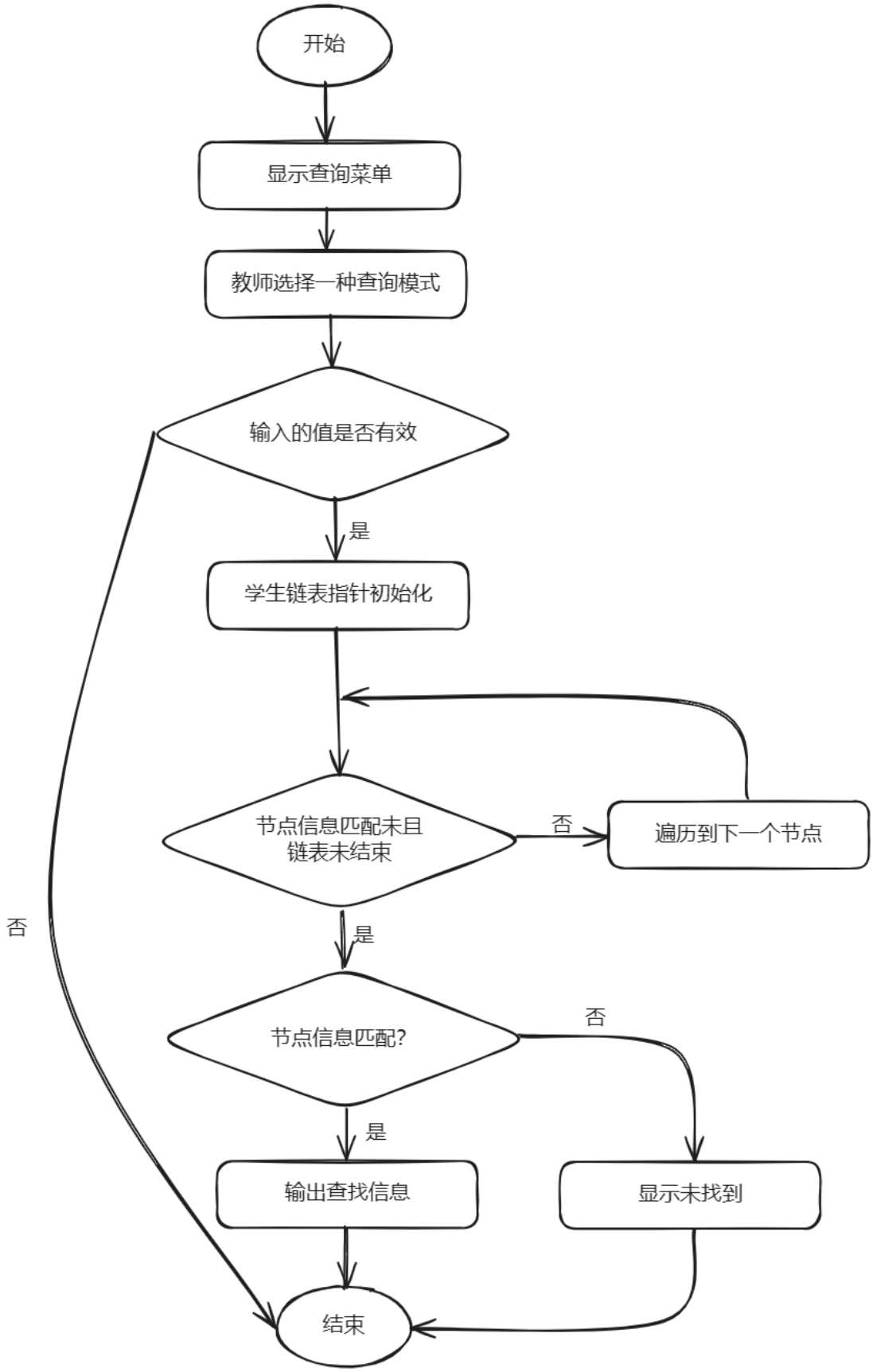
从第一个 Stulist 类的第一个节点开始，会将链表中的节点->getId()与要查找的 ID 比较，不一致时，p=p->next，与第二个数据对比，以此类推，直到找到相同的 ID 的节点，此时调用函数返回并输出关于该学生的所有信息。

为了代码的可读性和封装性，将此功能封装成一个函数，实现如下：

```
1.  Student *Stulist::checkIfExist(string idv) {
2.      Student *p = head->next;
3.      while (p) {
4.          if (p->getId() == idv)
5.              return p;
6.          p = p->next;
7.      }
8.      return nullptr;
9.  }
```

对于程序的编写，主要依靠三个函数进行实现此功能。分别是 findStudentByID()、findStudentsByName()、findStudentsByClass()。

流程图如下：



具体实现代码如下：

```
void Stulist::findStudentByID(string classmanagev) {
```

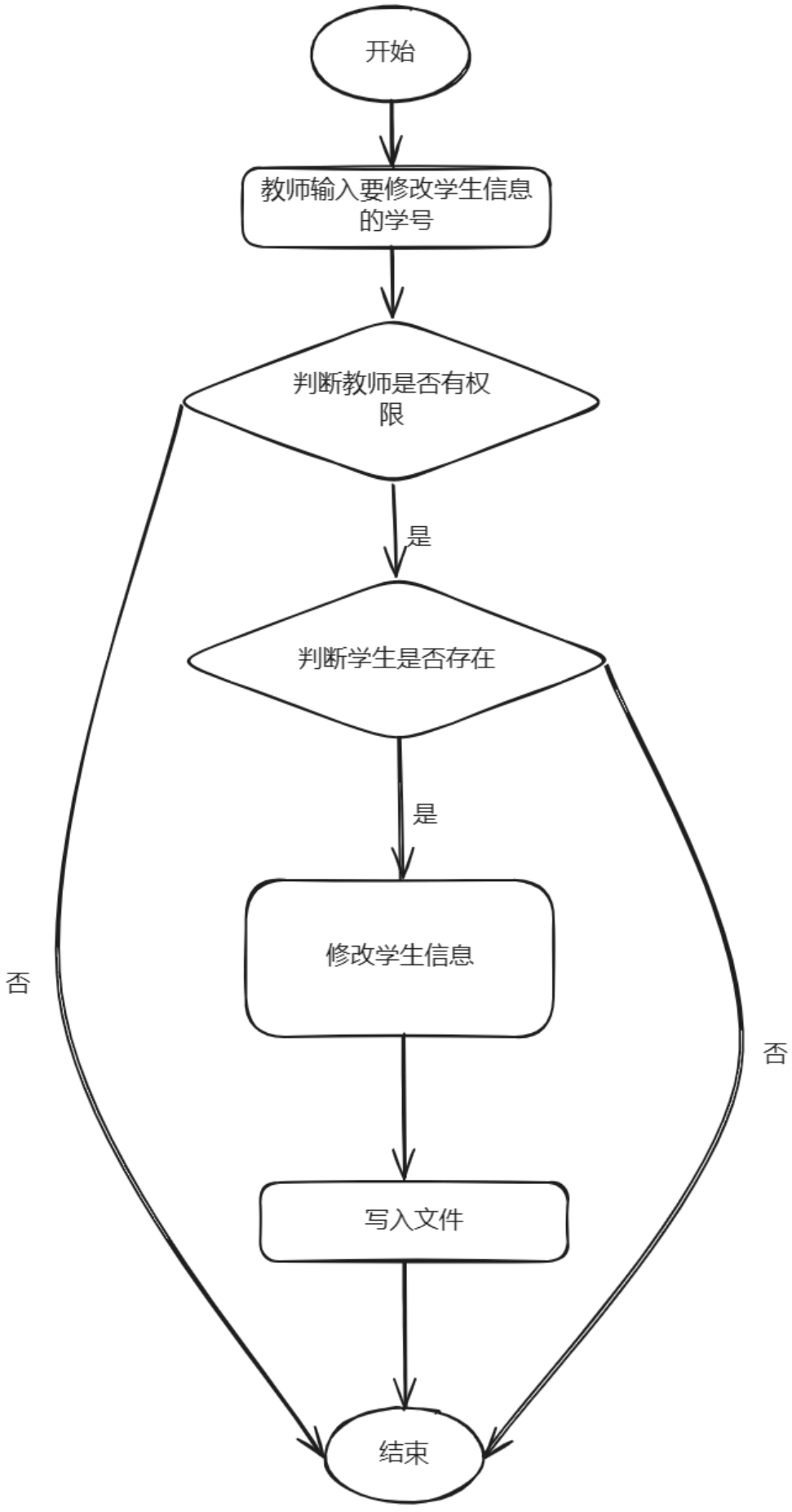
```
        string idv;
        cout << "请输入您要查找学生的学号:" << endl;
        cin >> idv;
        Student *stu = checkIfExist(idv);
        if (stu == nullptr) {
            cout << "很抱歉,并未找到该学生!" << endl;
            return;
        } else {
            if (stu->getClassName() == classmanagev) {
                cout << "----------------------------------" << endl;
                cout << "查询成功!学生信息为:" << endl;
                cout << "姓名:" << stu->getName() << endl;
                cout << "性别:" << stu->getGender() << endl;
                cout << "学号:" << stu->getId() << endl;
                cout << "班级:" << stu->getClassName() << endl;
                cout << "成绩(高数,程C,离散,大物):";
                const int *scores = stu->getScores();
                for (int i = 0; i < 4; ++i) {
                    cout << scores[i] << ' ';
                }
                cout << endl;
                cout << "总分:" << stu->getTotalScore() << endl;
                cout << "平均分:" << stu->getAverageScore() << endl;
                cout << "----------------------------------" << endl;
            } else {
                cout << "很抱歉,您无权操作!请联系管理员." << endl;
            }
        }
    }

    void Stulist::findStudentsByName(string classmanagev) {
        string namev;
        cout << "请输入您要查找学生的姓名(支持模糊查找):" << endl;
        cin >> namev;

        bool found = false;
        Student *p = head->next;
        while (p) {
            if (p->getName().find(namev) != string::npos && p->getClassName()
    == classmanagev) {
                found = true;
                cout << "姓名:" << p->getName() << endl;
                cout << "性别:" << p->getGender() << endl;
                cout << "学号:" << p->getId() << endl;
```

```
            cout << "班级:" << p->getClassName() << endl;
            cout << "成绩(高数,程C,离散,大物):";
            const int *scores = p->getScores();
            for (int i = 0; i < 4; ++i) {
                cout << scores[i] << ' ';
            }
            cout << endl;
            cout << "总分:" << p->getTotalScore() << endl;
            cout << "平均分:" << p->getAverageScore() << endl;
            cout << "----------------------------------" << endl;
        }
        p = p->next;
    }
    if (!found) {
        cout << "未找到符合条件的学生.请确认是否是您班级里的学生!" << endl;
        return;
    } else {
        cout << "查询成功!找到以上学生." << endl;
        return;
    }
}

void Stulist::findStudentsByClass(string classmanagev) {
    cout << "请注意,查询到的为您班级上的学生.如果未查到指定的学生,可能是因为该学生为其他班级上的或者该学生的信息并未填写提交!"
         << endl;
    cout << "----------------------------------------------------------------------------------" << endl;
    cout << "查询成功!找到以下学生." << endl;
    cout << "如果想要获取学生的详细信息,请使用学号或姓名查询!" << endl;

    Student *p = head->next;
    while (p) {
        if (p->getClassName() == classmanagev) {
            cout << "姓名          学号" << endl;
            cout << p->getName() << "          " << p->getId() << endl;
        }
        p = p->next;
    }
    cout << "----------------------------------------------------------------------------------" << endl;
}
```

- **学生信息的修改**

学生信息的修改，需要借助上一步的查找。即先查找出该账户所在的节点，再对该节点进行修改。若找不到则输出未找到该账户。找到了之后需要教师输入需要修改的信息。

流程图如下：


开始
教师输入要修改学生信息的学号
判断教师是否有权限
是
否
判断学生是否存在
是
否
修改学生信息
写入文件
结束

具体代码实现如下：

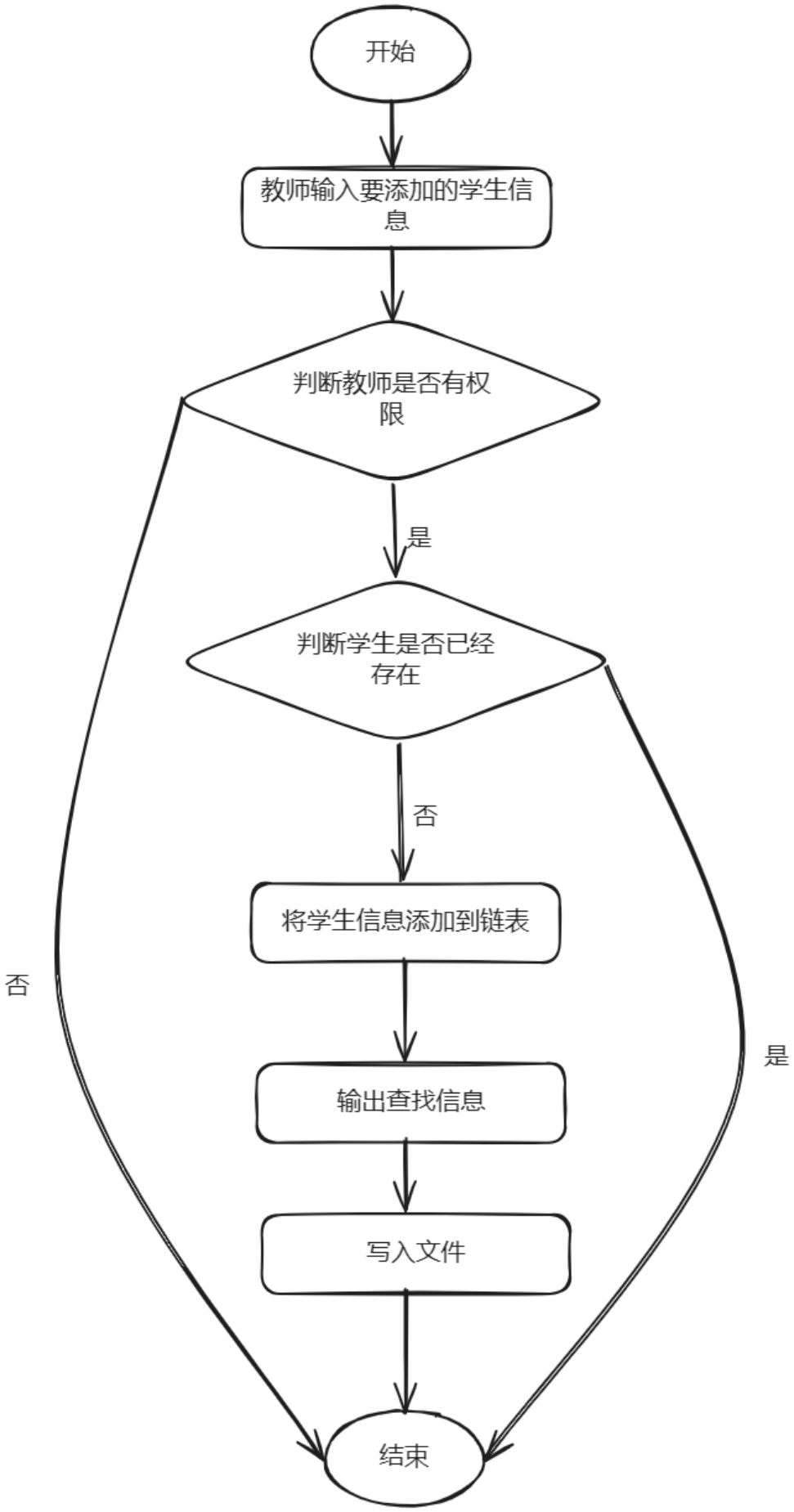
```
void Stulist::updateStudentByID(string classmanagev) {
    string idv;
    cout << "请输入您要修改学生信息的学号:" << endl;
    cin >> idv;
    Student *stu = checkIfExist(idv);
    if (stu == nullptr) {
        cout << "很抱歉，并未找到该学生!" << endl;
        return;
    } else {
        if (stu->getClassName() == classmanagev) {
            cout << "----------------------------------" << endl;
            cout << "当前学生信息为:" << endl;
            cout << "姓名:" << stu->getName() << endl;
            cout << "性别:" << stu->getGender() << endl;
            cout << "学号:" << stu->getId() << endl;
            cout << "班级:" << stu->getClassName() << endl;
            cout << "成绩(高数,程C,离散,大物):";
            const int *scores = stu->getScores();
            for (int i = 0; i < 4; ++i) {
                cout << scores[i] << ' ';
            }
            cout << endl;
            cout << "总分:" << stu->getTotalScore() << endl;
            cout << "平均分:" << stu->getAverageScore() << endl;
            cout << "----------------------------------" << endl;

            cout << "请输入修改后的信息(注意:如果信息不变请输入原来
的!):" << endl;
            string namev, genderv;
            int scoresv[4];
            cout << "姓名:" << endl;
            cin >> namev;
            cout << "性别:" << endl;
            cin >> genderv;
            if (!(genderv == "男" || genderv == "女")){
                cout << "性别输入无效!" << endl;
                return;
            }
            cout << "成绩(高数,程C,离散,大物):" << endl;
            for (int i = 0; i < 4; ++i) cin >> scoresv[i];

            stu->setName(namev);
            stu->setGender(genderv);
```

```
                stu->readScores(scoresv);

                write();
                cout << "学生信息修改成功!" << endl;
                cout << "姓名:" << stu->getName() << endl;
                cout << "性别:" << stu->getGender() << endl;
                cout << "学号:" << stu->getId() << endl;
                cout << "班级:" << stu->getClassName() << endl;
                cout << "成绩(高数,程 C,离散,大物):";
                for (int i = 0; i < 4; ++i) {
                    cout << scores[i] << ' ';
                }
                cout << endl;
                cout << "---------------------------------" << endl;
            } else {
                cout << "很抱歉,您无权操作!请联系管理员." << endl;
            }
        }
    }
```

- **学生信息的添加：**

添加学生信息，在 head->next 处增加一个节点。

流程图如下：


开始
教师输入要添加的学生信息
判断教师是否有权限
是
否
判断学生是否已经存在
否
是
将学生信息添加到链表
输出查找信息
写入文件
结束

具体代码实现如下：

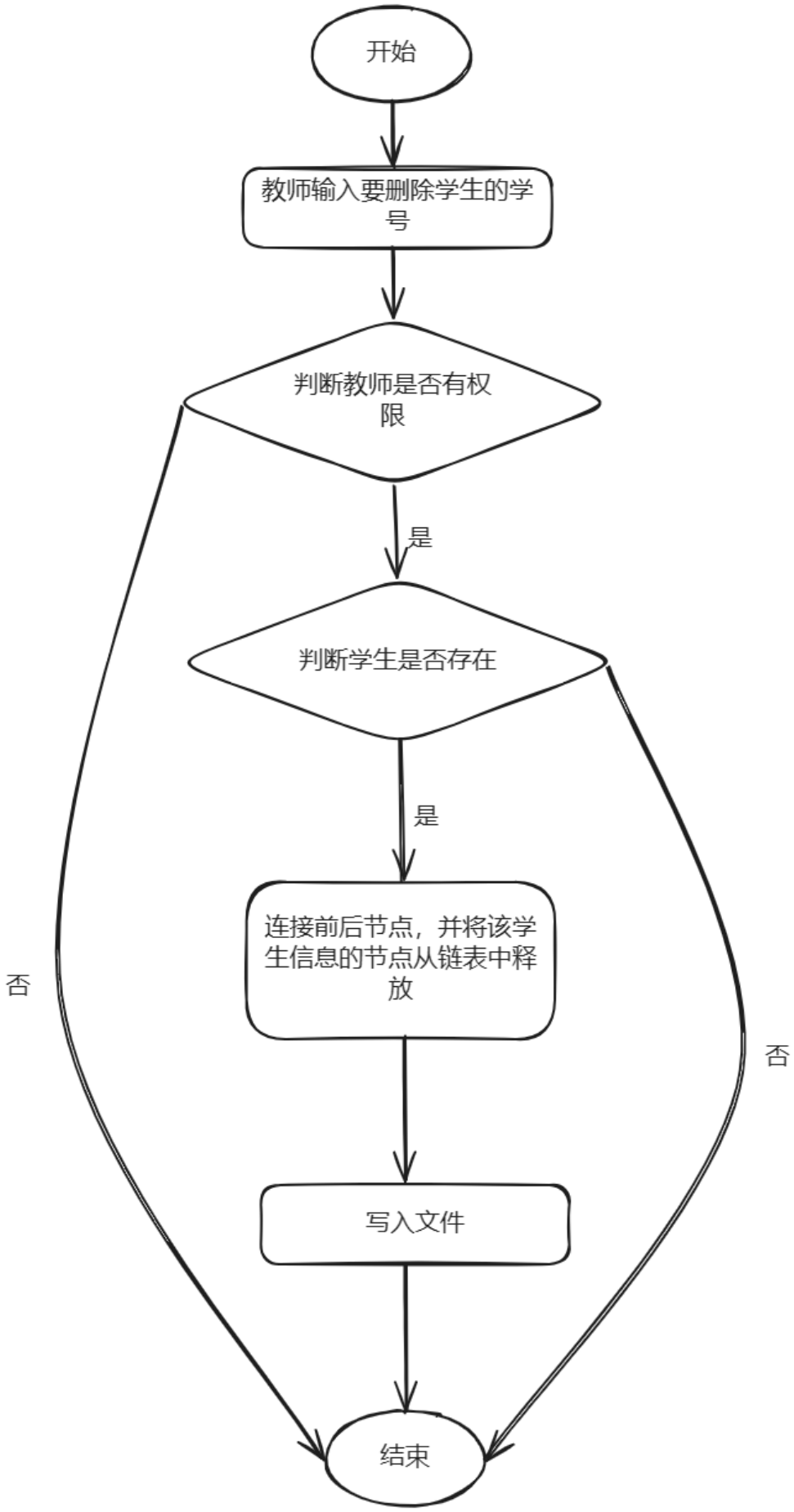
```
void Stulist::addStudentByTeacher(string classmanagev) {
    string namev, genderv, idv, classNamev;
    int scoresv[4];
    cout << "请输入您要添加学生的姓名:" << endl;
    cin >> namev;
    cout << "请输入您要添加学生的性别:" << endl;
    cin >> genderv;
    if (!(genderv == "男" || genderv == "女")){
        cout << "性别输入无效!" << endl;
        return;
    }
    cout << "请输入您要添加学生的学号:" << endl;
    cin >> idv;
    if (!(idv.length() == 12 && all_of(idv.begin(), idv.end(), ::isdigit))){
        cout << "学号输入无效!请重新输入(12 位数字)." << endl;
        return;
    }
    cout << "请输入您要添加学生的班级:" << endl;
    cin >> classNamev;
    cout << "请输入您要添加学生的成绩(四门分别是高数,程 C,离散,大物.请按顺序填
写!):" << endl;
    for (int i = 0; i < 4; ++i) cin >> scoresv[i];
    if (classNamev == classmanagev) {
        Student *stu = checkIfExist(idv);
        if (stu != nullptr) {
            cout << "您添加的学生已经存在!" << endl;
            return;
        }
        Student *p = new Student(namev, genderv, idv, classNamev);
        p->readScores(scoresv);
        p->next = head->next;
        head->next = p;
        size++;
        write();
        cout << "添加成功!" << endl;
        cout << "姓名:" << namev << endl;
        cout << "性别:" << genderv << endl;
        cout << "学号:" << idv << endl;
        cout << "班级:" << classNamev << endl;
        cout << "成绩(高数,程 C,离散,大物):";
        for (int i = 0; i < 4; ++i) {
            cout << scoresv[i] << ' ';
        }
```

```
            cout << endl;
            cout << "----------------------------------" << endl;
        } else {
            cout << "很抱歉,您无权操作!请联系管理员." << endl;
        }
    }
```

- **学生信息的删除：**

删除目标学生信息的节点的思路和修改账户类似，先通过目标信息检索查找出该学生信息所在的节点，然后进行删除节点操作。即将 pre 前指针与 next 后指针相连，将当前节点略过，并释放，即可实现删除节点的效果。

流程图如下：


开始
教师输入要删除学生的学号
判断教师是否有权限
是
判断学生是否存在
是
连接前后节点，并将该学生信息的节点从链表中释放
写入文件
结束
否
否

具体代码实现如下：

```
void Stulist::deleteStudentByID(string classmanagev) {
    string idv;
    cout << "请输入您要删除学生信息的学号:" << endl;
    cin >> idv;

    Student *current = head->next;
    Student *pre = nullptr;
    while (current != nullptr) {
        if (current->getId() == idv && classmanagev != current->getClassName(
)) {
            cout << "很抱歉,您无权操作!请联系管理员." << endl;
            return;
        }
        if (current->getId() == idv && classmanagev == current->getClassName(
)) {
            if (pre == nullptr) {
                head->next = current->next;
            } else {
                pre->next = current->next;
            }
            delete current;
            write();
            cout << "成功删除学生信息!" << endl;
            return;
        }
        pre = current;
        current = current->next;
    }

    cout << "很抱歉，并未找到该学生!" << endl;
}
```

- **学生成绩的统计**

由于每个老师仅拥有对自己班级学生信息管理的权限，所以在统计学生成绩时，教师仅能查看自己班级的学生人数、各科成绩平均分、各科及格人数和平均分。

在统计学生成绩时，需要先将本班级的学生从链表中选取出来。这里先创建了一个新的链表 classStudents，然后对原链表进行遍历，匹配相同班级的学生，放入 classStudents 链表中，最后返回链表。

具体代码实现如下：

```
Student *Stulist::findTheSameClassStudents(string classmanagev) {
    // 创建一个新的链表来存储给定班级的学生
    Student *classStudents = nullptr;
    // 遍历链表，将给定班级的学生添加到新链表中
```

```
    Student *p = head->next;
    while (p) {
        if (p->getClassName() == classmanagev) {
            Student *newStudent = new Student(*p);
            newStudent->next = classStudents;
            classStudents = newStudent;
        }
        p = p->next;
    }
    return classStudents;
}
```

在获取到新链表后，进行统计。具体提代码实现如下：

```
void Stulist::countStudentsScore(string classmanagev) {

    Student *classStudents = findTheSameClassStudents(classmanagev);

    // 插入排序按总分降序排序
    Student *sortedList = nullptr;
    while (classStudents) {
        // 从未排序链表中移除一个节点
        Student *current = classStudents;
        classStudents = classStudents->next;
        // 将节点插入到已排序链表中的正确位置
        if (!sortedList || current->getTotalScore() > sortedList->getTotal
Score()) {
            // 将节点插入到已排序链表的开头
            current->next = sortedList;
            sortedList = current;
        } else {
            // 在已排序链表中找到插入位置
            Student *temp = sortedList;
            while (temp->next && temp->next->getTotalScore() > current->ge
tTotalScore()) {
                temp = temp->next;
            }
            // 将节点插入到已排序链表的中间
            current->next = temp->next;
            temp->next = current;
        }
    }

    cout << "统计成功!" << endl << classmanagev << "班学生成绩:" << endl;
    cout << "----------------------------------------------------------------
--------------------" << endl;
```

```
    int number = 1;
    double totalAverage = 0.0;
    int numAbove60[4] = {0};
    double totalSubjectScores[4] = {0};
    while (sortedList) {
        const int *scores = sortedList->getScores();
        for (int i = 0; i < 4; ++i) {
            totalSubjectScores[i] += scores[i];
            if (scores[i] > 60) {
                numAbove60[i]++;
            }
        }
        totalAverage += sortedList->getTotalScore();

        Student *temp = sortedList;
        sortedList = sortedList->next;
        delete temp;
        number++;
    }
    // 输出班级总人数
    cout << "班级总人数:" << number - 1 << endl;
    cout << endl;
    // 输出各科成绩平均分
    cout << "各科成绩平均分:" << endl;
    const char *subjectNames[4] = {"高数", "程 C", "离散", "大物"};
    for (int i = 0; i < 4; ++i) {
        cout << subjectNames[i] << ": " << totalSubjectScores[i] / ((numbe
r - 1) * 1.0) << endl;
    }
    cout << endl;
    // 输出各科成绩超过 60 分的人数
    cout << "各科及格的人数:" << endl;
    for (int i = 0; i < 4; ++i) {
        cout << subjectNames[i] << ": " << numAbove60[i] << endl;
    }
    cout << endl;
    // 输出总分的平均分
    cout << "班级平均分:" << totalAverage / ((number - 1) * 1.0) << endl;
    cout << "----------------------------------------------------------
--------------------" << endl;
}
```

- **学生成绩的排序**

  先获取本班级下的学生链表。

排序我采用的是插入排序算法(以平均分降序为例)。首先，创建了一个空的已排序链表。然后，循环遍历未排序的学生链表。在每次迭代中，从未排序链表中移除一个学生节点，并将其插入到已排序链表的正确位置。如果已排序链表为空，或者当前节点的平均分高于已排序链表的第一个节点，那么当前节点将成为新的已排序链表的头节点。如果当前节点的平均分介于已排序链表中的某两个节点之间，那么它将被插入到这两个节点之间。

具体代码实现如下：

```
void Stulist::sortStudentsByID(string classmanagev) {

    Student *classStudents = findTheSameClassStudents(classmanagev);

    // 插入排序对链表中的学生按学号升序排序
    Student *sortedList = nullptr;
    while (classStudents) {
        // 从未排序链表中移除一个节点
        Student *current = classStudents;
        classStudents = classStudents->next;
        // 将节点插入到已排序链表中的正确位置
        if (!sortedList || current->getId() < sortedList->getId()) {
            // 将节点插入到已排序链表的开头
            current->next = sortedList;
            sortedList = current;
        } else {
            // 在已排序链表中找到插入位置
            Student *temp = sortedList;
            while (temp->next && temp->next->getId() < current->getId()) {

                temp = temp->next;
            }
            // 将节点插入到已排序链表的中间
            current->next = temp->next;
            temp->next = current;
        }
    }

    cout << "排序成功!(按学号升序排序)" << endl << classmanagev << "班学生列表:" << endl;
    cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)" << endl;
    cout << "----------------------------------------------------------------------" << endl;
    int number = 1;
    while (sortedList) {
```

```
            cout << number++ << " " << sortedList->getName() << " " << sortedL
ist->getGender() << " " << sortedList->getId()
                 << " ";
            const int *scores = sortedList->getScores();
            for (int i = 0; i < 4; ++i) {
                cout << scores[i] << ' ';
            }
            cout << sortedList->getTotalScore() << " " << sortedList->getAvera
geScore() << endl;
            // 移动到下一个节点并释放当前节点的内存
            Student *temp = sortedList;
            sortedList = sortedList->next;
            delete temp;
        }
        cout << "-------------------------------------------------------------
---------------------" << endl;
    }

    void Stulist::sortStudentsBySubjectScore(string classmanagev) {

        Student *classStudents = findTheSameClassStudents(classmanagev);

        int index;
        cout << "请输入您要根据哪门成绩进行排序(1:高数 2:程C 3:离散 4:大
物 )" << endl;
        cin >> index;
        // 插入排序按指定科目成绩降序排序
        Student *sortedList = nullptr;
        while (classStudents) {
            // 从未排序链表中移除一个节点
            Student *current = classStudents;
            classStudents = classStudents->next;
            // 将节点插入到已排序链表中的正确位置
            if (!sortedList || current->getScores()[index - 1] > sortedList->g
etScores()[index - 1]) {
                // 将节点插入到已排序链表的开头
                current->next = sortedList;
                sortedList = current;
            } else {
                // 在已排序链表中找到插入位置
                Student *temp = sortedList;
                while (temp->next && temp->next->getScores()[index - 1] > curr
ent->getScores()[index - 1]) {
                    temp = temp->next;
```

```
            }
            // 将节点插入到已排序链表的中间
            current->next = temp->next;
            temp->next = current;
        }
    }

    cout << "排序成功!(按单科成绩降序排序)" << endl << classmanagev << "班学生列表:" << endl;
    cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)" << endl;
    cout << "------------------------------------------------------------------------" << endl;
    int number = 1;
    while (sortedList) {
        cout << number++ << " " << sortedList->getName() << " " << sortedList->getGender() << " " << sortedList->getId()
             << " ";
        const int *scores = sortedList->getScores();
        for (int i = 0; i < 4; ++i) {
            cout << scores[i] << ' ';
        }
        cout << sortedList->getTotalScore() << " " << sortedList->getAverageScore() << endl;
        // 移动到下一个节点并释放当前节点的内存
        Student *temp = sortedList;
        sortedList = sortedList->next;
        delete temp;
    }
    cout << "------------------------------------------------------------------------" << endl;
}

void Stulist::sortStudentsByTotalScore(string classmanagev) {

    Student *classStudents = findTheSameClassStudents(classmanagev);
    // 插入排序按总分降序排序
    Student *sortedList = nullptr;
    while (classStudents) {
        // 从未排序链表中移除一个节点
        Student *current = classStudents;
        classStudents = classStudents->next;
        // 将节点插入到已排序链表中的正确位置
```

```
        if (!sortedList || current->getTotalScore() > sortedList->getTotal
Score()) {
            // 将节点插入到已排序链表的开头
            current->next = sortedList;
            sortedList = current;
        } else {
            // 在已排序链表中找到插入位置
            Student *temp = sortedList;
            while (temp->next && temp->next->getTotalScore() > current->ge
tTotalScore()) {
                temp = temp->next;
            }
            // 将节点插入到已排序链表的中间
            current->next = temp->next;
            temp->next = current;
        }
    }

    cout << "排序成功!(按总分降序排序)" << endl << classmanagev << "班学生列
表:" << endl;
    cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大
物||总分||平均分)" << endl;
    cout << "----------------------------------------------------------------
---------------------" << endl;
    int number = 1;
    while (sortedList) {
        cout << number++ << " " << sortedList->getName() << " " << sortedL
ist->getGender() << " " << sortedList->getId()
             << " ";
        const int *scores = sortedList->getScores();
        for (int i = 0; i < 4; ++i) {
            cout << scores[i] << ' ';
        }
        cout << sortedList->getTotalScore() << " " << sortedList->getAvera
geScore() << endl;
        // 移动到下一个节点并释放当前节点的内存
        Student *temp = sortedList;
        sortedList = sortedList->next;
        delete temp;
    }
    cout << "----------------------------------------------------------------
---------------------" << endl;
}

```

```
void Stulist::sortStudentsByAverageScore(string classmanagev) {

    Student *classStudents = findTheSameClassStudents(classmanagev);

    // 插入排序按平均分降序排序
    Student *sortedList = nullptr;
    while (classStudents) {
        // 从未排序链表中移除一个节点
        Student *current = classStudents;
        classStudents = classStudents->next;
        // 将节点插入到已排序链表中的正确位置
        if (!sortedList || current->getAverageScore() > sortedList->getAve
rageScore()) {
            // 将节点插入到已排序链表的开头
            current->next = sortedList;
            sortedList = current;
        } else {
            // 在已排序链表中找到插入位置
            Student *temp = sortedList;
            while (temp->next && temp->next->getAverageScore() > current->
getAverageScore()) {
                temp = temp->next;
            }
            // 将节点插入到已排序链表的中间
            current->next = temp->next;
            temp->next = current;
        }
    }

    cout << "排序成功!(按平均分降序排序)" << endl << classmanagev << "班学生
列表:" << endl;
    cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大
物||总分||平均分)" << endl;
    cout << "----------------------------------------------------------------
--------------------" << endl;
    int number = 1;
    while (sortedList) {
        cout << number++ << " " << sortedList->getName() << " " << sortedL
ist->getGender() << " " << sortedList->getId()
             << " ";
        const int *scores = sortedList->getScores();;
        for (int i = 0; i < 4; ++i) {
            cout << scores[i] << ' ';
        }
```

```
            cout << sortedList->getTotalScore() << " " << sortedList->getAvera
    geScore() << endl;
            // 移动到下一个节点并释放当前节点的内存
            Student *temp = sortedList;
            sortedList = sortedList->next;
            delete temp;
        }
        cout << "-------------------------------------------------------------
    ---------------------" << endl;
    }
```

（3）交互界面以及登录菜单的实现

系统运行开始的界面如图 4 所示：

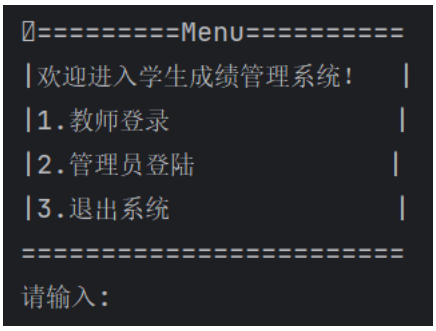


图 4 开始登录界面

主要通过选择结构和循环结构实现界面的前进和后退。例如，第一个教师登录界面出现 7 个选择：1.添加学生信息，2.查询学生信息，3.排序学生成绩，4.统计学生成绩，5.修改学生信息，6.删除学生信息，7.返回上级菜单。使用 switch case 分别实现，选择之后转到下一个界面。如果该功能已完成或者用户输入了无效值，系统会执行 goto 返回到上一级菜单。

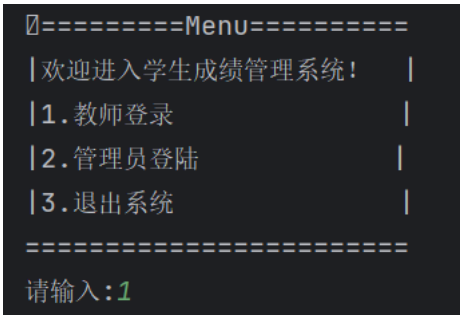
四、 实验调试、测试、运行记录及分析

系统在调试测试过程中遇到若干问题，不过经过仔细反复的检查已经消除各种 bug。

主要的测试经过如下：

4.1 教师模块

4.1.1 进入系统选择 1 教师登录



根据 Manager.txt 存储的账户密码完成登录。

```
请输入您的身份(1:教师 2:管理员):
1
请输入账号:
zjut001
请输入密码:
123456
```

如果连续五次密码输入错误，该账号会被锁定。

```
请输入您的身份(1:教师 2:管理员):
1
请输入账号:
zjut004
请输入密码:
12345
账号或密码错误! 请注意,您还有4次机会!
请输入账号:
zjut004
请输入密码:
12345
账号或密码错误! 请注意,您还有3次机会!
请输入账号:
zjut004
请输入密码:
12345
账号或密码错误! 请注意,您还有2次机会!
请输入账号:
zjut004
请输入密码:
12345
账号或密码错误! 请注意,您还有1次机会!
请输入账号:
zjut004
请输入密码:
12345
账号或密码错误! 请注意,您还有0次机会!
您已达到最大尝试次数,帐号已被锁定.请联系管理员!
```

再次登陆时，会显示：

```
请输入您的身份(1:教师 2:管理员):
1
请输入账号:
zjut004
请输入密码:
123456
您的账号已被锁定!请联系管理员.
```

4.1.2 教师获取功能菜单

```
=======教师功能菜单=======
|1. 添加学生信息          |
|2. 查询学生信息          |
|3. 排序学生成绩          |
|4. 统计学生成绩          |
|5. 修改学生信息          |
|6. 删除学生信息          |
|7. 返回上级菜单          |
========================
请输入:
```

4.1.3 教师添加学生信息

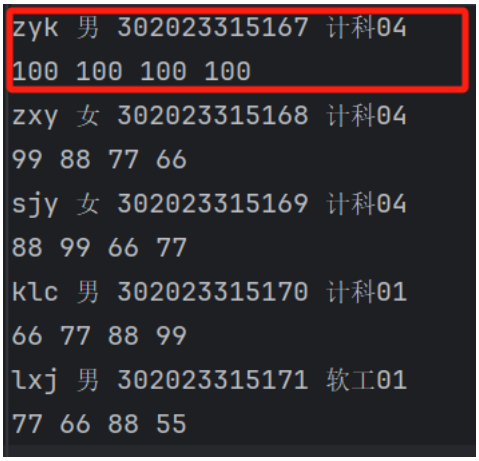
输入要添加的学生信息（只能添加本班学生）。

```
请输入您要添加学生的姓名:
zyk
请输入您要添加学生的性别:
男
请输入您要添加学生的学号:
302023315167
请输入您要添加学生的班级:
计科04
请输入您要添加学生的成绩(四门分别是高数,程C,离散,大物.请按顺序填写!):
100 100 100 100
```

输出新添加的学生信息。

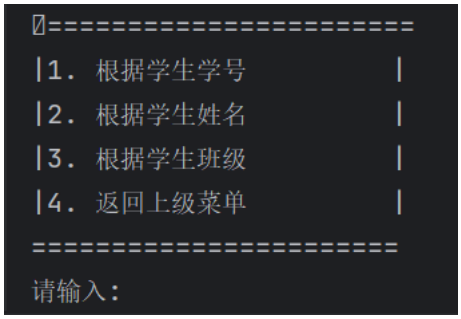
```
添加成功!
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):100 100 100 100
--------------------------------
```

可以看到 Student.txt 中已经有相应的数据了。



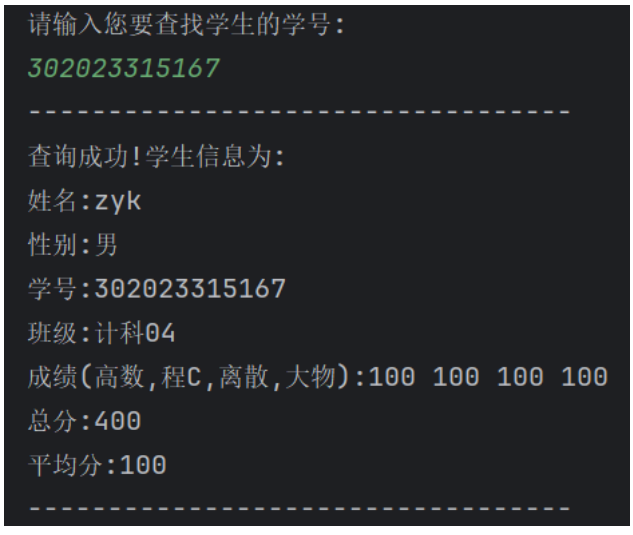
4.1.4 教师查询学生信息（本班学生）

可以选择查询依据。



4.1.4.1 根据学生学号

教师可以根据学生的学号进行精确查找。



4.1.4.2 根据学生姓名

教师可以根据学生的姓名进行模糊查找。

```
请输入您要查找学生的姓名(支持模糊查找):
z
姓名:zxy
性别:女
学号:302023315168
班级:计科04
成绩(高数,程C,离散,大物):99 88 77 66
总分:330
平均分:82.5
------------------------------------
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):100 100 100 100
总分:400
平均分:100
------------------------------------
```

4.1.4.3 根据学生班级

由于教师仅拥有对自己班级的学生信息的查询权限，故会返回本班级所有学生信息。

```
请注意,查询到的为您班级上的学生.如果未查到指定的学生,可能是因为该学生为其他班级上的或者该学生的信息并未填写提交!
-----------------------------------------------------------------------------------------
查询成功!找到以下学生.
如果想要获取学生的详细信息,请使用学号或姓名查询!
姓名            学号
sjy          302023315169
姓名            学号
zxy          302023315168
姓名            学号
zyk          302023315167
-----------------------------------------------------------------------------------------
```

4.1.5 教师排序学生成绩（本班学生）

可以选择排序依据。

```
|1. 根据学生学号          |
|2. 根据单科成绩          |
|3. 根据平均成绩          |
|4. 根据学生总分          |
|5. 返回上级菜单          |
========================
请输入:
```

4.1.5.1 根据学生学号

教师可以根据学生学号进行排序（升序）。

```
排序成功!(按学号升序排序)
计科04班学生列表:
(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
-------------------------------------------------------------------------------
1 zyk 男 302023315167 100 100 100 100 400 100
2 zxy 女 302023315168 99 88 77 66 330 82.5
3 sjy 女 302023315169 88 99 66 77 330 82.5
-------------------------------------------------------------------------------
```

4.1.5.2 根据单科成绩

教师可以选择一门成绩作为依据进行排序（降序）。

```
请输入您要根据哪门成绩进行排序(1:高数 2:程C 3:离散 4:大物 )
1
排序成功!(按单科成绩降序排序)
计科04班学生列表:
(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
-------------------------------------------------------------------------------
1 zyk 男 302023315167 100 100 100 100 400 100
2 zxy 女 302023315168 99 88 77 66 330 82.5
3 sjy 女 302023315169 88 99 66 77 330 82.5
-------------------------------------------------------------------------------
```

4.1.5.3 根据平均成绩

教师可以根据学生四门课程的平均成绩进行排序（降序）。

```
排序成功!(按平均分降序排序)
计科04班学生列表:
(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
-------------------------------------------------------------------------------
1 zyk 男 302023315167 100 100 100 100 400 100
2 sjy 女 302023315169 88 99 66 77 330 82.5
3 zxy 女 302023315168 99 88 77 66 330 82.5
-------------------------------------------------------------------------------
```

4.1.5.4 根据学生总分

教师可以根据学生的总成绩进行排序（降序）。

```
排序成功!(按总分降序排序)
计科04班学生列表:
(名次||姓名||性别||学号||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
--------------------------------------------------------------------------------
1 zyk 男 302023315167 100 100 100 100 400 100
2 sjy 女 302023315169 88 99 66 77 330 82.5
3 zxy 女 302023315168 99 88 77 66 330 82.5
--------------------------------------------------------------------------------
```

4.1.6 教师统计学生成绩（本班学生）

教师可以获取本班级的学生人数，各科平均分，各科 60 分以上的人数，平均分。

```
统计成功!
计科04班学生成绩:
--------------------------------------------------------------------------------
班级总人数:3

各科成绩平均分:
高数: 95.6667
程C: 95.6667
离散: 81
大物: 81

各科及格的人数:
高数: 3
程C: 3
离散: 3
大物: 3

班级平均分:353.333
--------------------------------------------------------------------------------
```

4.1.7 教师修改学生信息（本班学生）

教师先根据学生的学号查询到对应的信息，然后输入修改后的数据（如果此数据不修改，需要输入原来的数据）。

```
请输入您要修改学生信息的学号:
302023315167
-----------------------------------
当前学生信息为:
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):100 100 100 100
总分:400
平均分:100
-----------------------------------
请输入修改后的信息(注意:如果信息不变请输入原来的!):
姓名:
zyk
性别:
男
成绩(高数,程C,离散,大物):
99 100 100 99
学生信息修改成功!
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):99 100 100 99
-----------------------------------
```

可以看到数据已经更新到 Student.txt 文件中了。

```
lxj 男 302023315171 软工01
77 66 88 55
klc 男 302023315170 计科01
66 77 88 99
sjy 女 302023315169 计科04
88 99 66 77
zxy 女 302023315168 计科04
99 88 77 66
zyk 男 302023315167 计科04
99 100 100 99
```

4.1.8 删除学生信息（本班学生）

教师可以根据学生的学号删除该学生的信息。

```
请输入您要删除学生信息的学号:
302023315167
成功删除学生信息!
```

可以看到数据已经更新到 Student.txt 文件中了。

```
zxy 女 302023315168 计科04
99 88 77 66
sjy 女 302023315169 计科04
88 99 66 77
klc 男 302023315170 计科01
66 77 88 99
lxj 男 302023315171 软工01
77 66 88 55
```

4.2 管理员模块

4.2.1 进入系统选择 2 管理员登录

```
=========Menu==========
|欢迎进入学生成绩管理系统!   |
|1.教师登录              |
|2.管理员登陆            |
|3.退出系统              |
=======================
请输入:1
```

根据 Manager.txt 存储的账户密码完成登录。

```
请输入您的身份(1:教师 2:管理员):
2
请输入账号:
ZJUT
请输入密码:
123456
```

如果该账号不是管理员账号，那么将会输出错误信息并返回上一级菜单。

```
请输入您的身份(1:教师 2:管理员):
2
请输入账号:
zjut001
请输入密码:
123456
很抱歉,您不是管理员!
```

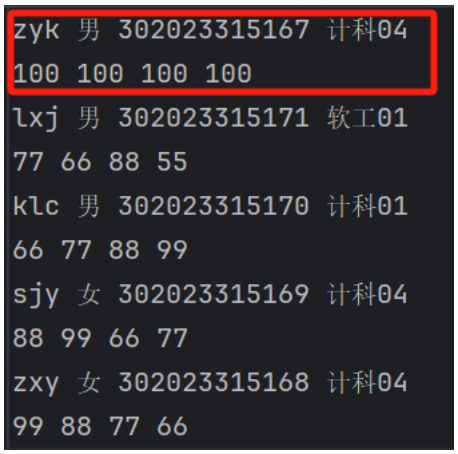
4.2.2 管理员获取功能菜单

```
□=====管理员功能菜单=====
|1.  添加学生信息        |
|2.  查询学生信息        |
|3.  排序学生成绩        |
|4.  统计学生成绩        |
|5.  修改学生信息        |
|6.  删除学生信息        |
|7.  添加教师账号        |
|8.  删除教师账号        |
|9.  重置账号密码        |
|10. 修改账号状态        |
|11. 返回上级菜单        |
========================
请输入:
```

4.2.3 管理员添加学生信息

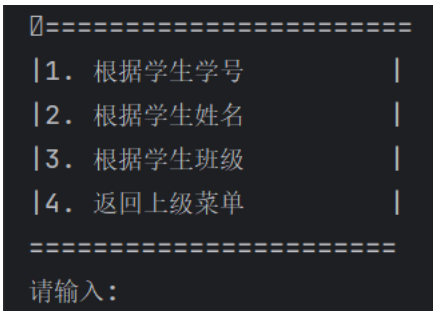
```
请输入您要添加学生的姓名:
zyk
请输入您要添加学生的性别:
男
请输入您要添加学生的学号:
302023315167
请输入您要添加学生的班级:
计科04
请输入您要添加学生的成绩(四门分别是高数,程C,离散,大物.请按顺序填写!):
100 100 100 100
添加成功!
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):100 100 100 100
----------------------------------
```

可以看 Student.txt 中已经有数据了。



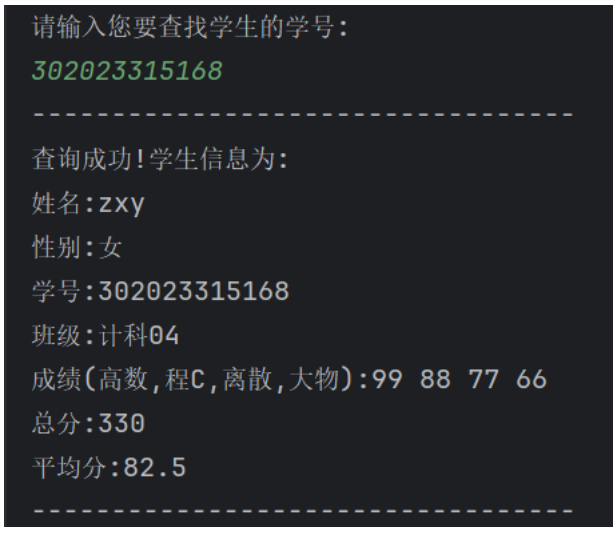
4.2.4 管理员查询学生信息

可以选择查询依据。



4.2.4.1 根据学生学号

管理员可以根据学生的学号进行精确查找。



4.2.4.2 根据学生姓名

管理员可以根据学生的姓名进行模糊查找。

```
请输入您要查找学生的姓名(支持模糊查找):
z
姓名:zxy
性别:女
学号:302023315168
班级:计科04
成绩(高数,程C,离散,大物):99 88 77 66
总分:330
平均分:82.5
-----------------------------------
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):100 100 100 100
总分:400
平均分:100
-----------------------------------
```

4.2.4.3 根据学生班级

管理员可以模糊查找已有的班级。

```
请输入您要查找学生的班级(支持模糊查找):
计
查询成功!找到以下班级:
如果想要获取学生的详细信息,请使用学号或姓名查询!
------------------------------
班级: 计科01
姓名            学号
klc          302023315170
------------------------------
班级: 计科04
姓名            学号
zyk          302023315167
sjy          302023315169
zxy          302023315168
------------------------------
```

4.2.5 管理员排序学生成绩

可以选择排序依据。

```
|======================
|1. 根据学生学号        |
|2. 根据单科成绩        |
|3. 根据平均成绩        |
|4. 根据学生总分        |
|5. 返回上级菜单        |
========================
请输入:
```

4.2.5.1 根据学生学号

管理员可以根据学生学号进行排序（升序）。

```
排序成功!(按学号升序排序)
(名次||姓名||性别||学号||班级||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
--------------------------------------------------------------------------------
1 zyk 男 302023315167 计科04 100 100 100 100 400 100
2 zxy 女 302023315168 计科04 99 88 77 66 330 82.5
3 sjy 女 302023315169 计科04 88 99 66 77 330 82.5
4 klc 男 302023315170 计科01 66 77 88 99 330 82.5
5 lxj 男 302023315171 软工01 77 66 88 55 286 71.5
--------------------------------------------------------------------------------
```

4.2.5.2 根据单科成绩

管理员可以选择一门成绩作为依据进行排序（降序）。

```
请输入您要根据哪门成绩进行排序(1:高数 2:程C 3:离散 4:大物 )
1
排序成功!(按单科成绩降序排序)
(名次||姓名||性别||学号||班级||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
--------------------------------------------------------------------------------
1 zyk 男 302023315167 计科04 100 100 100 100 400 100
2 zxy 女 302023315168 计科04 99 88 77 66 330 82.5
3 sjy 女 302023315169 计科04 88 99 66 77 330 82.5
4 lxj 男 302023315171 软工01 77 66 88 55 286 71.5
5 klc 男 302023315170 计科01 66 77 88 99 330 82.5
--------------------------------------------------------------------------------
```

4.2.5.3 根据平均成绩

管理员可以根据学生四门课程的平均成绩进行排序（降序）。

```
排序成功!(按平均分降序排序)
(名次||姓名||性别||学号||班级||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
--------------------------------------------------------------------------------
1 zyk 男 302023315167 计科04 100 100 100 100 400 100
2 zxy 女 302023315168 计科04 99 88 77 66 330 82.5
3 sjy 女 302023315169 计科04 88 99 66 77 330 82.5
4 klc 男 302023315170 计科01 66 77 88 99 330 82.5
5 lxj 男 302023315171 软工01 77 66 88 55 286 71.5
--------------------------------------------------------------------------------
```

4.2.5.4 根据学生总分

管理员可以根据学生的总成绩进行排序（降序）。

```
排序成功!(按总分降序排序)
(名次||姓名||性别||学号||班级||成绩.高数||成绩.程C||成绩.离散||成绩.大物||总分||平均分)
--------------------------------------------------------------------------------
1 zyk 男 302023315167 计科04 100 100 100 100 400 100
2 zxy 女 302023315168 计科04 99 88 77 66 330 82.5
3 sjy 女 302023315169 计科04 88 99 66 77 330 82.5
4 klc 男 302023315170 计科01 66 77 88 99 330 82.5
5 lxj 男 302023315171 软工01 77 66 88 55 286 71.5
--------------------------------------------------------------------------------
```

4.2.6 管理员统计学生成绩

管理员可以获取各个班级的排名，学生人数，各科平均分，各科 60 分以上的人数，平均分。

```
统计成功!
-----------------------------
班级排名: 1. 计科04  平均分: 353.333
班级排名: 2. 计科01  平均分: 330
班级排名: 3. 软工01  平均分: 286
-----------------------------
计科01班学生成绩:
班级总人数:1

各科成绩平均分:
高数: 66
程C: 77
离散: 88
大物: 99

各科及格的人数:
高数: 1
程C: 1
离散: 1
大物: 1

班级平均分:330
-----------------------------
计科04班学生成绩:
班级总人数:3

各科成绩平均分:
高数: 95.6667
程C: 95.6667
离散: 81
大物: 81
```

```
各科及格的人数:
高数: 3
程C: 3
离散: 3
大物: 3

班级平均分:353.333
------------------------------
软工01班学生成绩:
班级总人数:1

各科成绩平均分:
高数: 77
程C: 66
离散: 88
大物: 55

各科及格的人数:
高数: 1
程C: 1
离散: 1
大物: 0

班级平均分:286
------------------------------
```

4.2.7 管理员修改学生信息

管理员先根据学生的学号查询到对应的信息，然后输入修改后的数据（如果此数据不修改，需要输入原来的数据）。

```
请输入您要修改学生信息的学号:
302023315167
-----------------------------------
当前学生信息为:
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):100 100 100 100
总分:400
平均分:100
-----------------------------------
请输入修改后的信息(注意:如果信息不变请输入原来的!):
姓名:
zyk
性别:
男
班级:
计科04
成绩(高数,程C,离散,大物):
100 99 99 100
学生信息修改成功!
姓名:zyk
性别:男
学号:302023315167
班级:计科04
成绩(高数,程C,离散,大物):100 99 99 100
-----------------------------------
```

可以看到数据已经更新到 Student.txt 文件中了。

```
zyk 男 302023315167 计科04
100 99 99 100
lxj 男 302023315171 软工01
77 66 88 55
klc 男 302023315170 计科01
66 77 88 99
sjy 女 302023315169 计科04
88 99 66 77
zxy 女 302023315168 计科04
99 88 77 66
```

4.2.8 删除学生信息

管理员可以根据学生的学号删除该学生的信息。

```
请输入您要删除学生信息的学号:
302023315167
成功删除学生信息!
```

可以看到数据已经更新到 Student.txt 文件中了。

```
lxj 男 302023315171 软工01
77 66 88 55
klc 男 302023315170 计科01
66 77 88 99
sjy 女 302023315169 计科04
88 99 66 77
zxy 女 302023315168 计科04
99 88 77 66
```

4.2.9 添加教师账号

管理员可以根据需求添加教师账号。

```
请输入您要添加教师账号的用户名:
zjut004
请输入您要添加教师账号的密码:
123456
请输入您要添加教师账号的权限:
数媒01
添加成功!
```

可以看到数据已经更新到 Manager.txt 文件中了。

```
zjut004 123456 1 1 数媒01
ZJUT 123456 2 1 0
zjut003 123456 1 1 网工01
zjut002 123456 1 1 软工01
zjut001 123456 1 1 计科04
```

4.2.10 删除教师账号

管理员可以根据需求删除教师账号。

```
请输入您要删除教师账号的用户名:
zjut004
删除成功!
```

可以看到数据已经更新到 Manager.txt 文件中了。

```
ZJUT 123456 2 1 0
zjut003 123456 1 1 网工01
zjut002 123456 1 1 软工01
zjut001 123456 1 1 计科04
```

4.2.11 重置账号密码

管理员可以重置账号密码。

```
请输入您要重置教师账号的用户名:
zjut001
请输入新密码:
12345678
请再次输入密码:
12345678
重置成功!
```

可以看到数据已经更新到 Manager.txt 文件中了。

```
ZJUT 123456 2 1 0
zjut003 123456 1 1 网工01
zjut002 123456 1 1 软工01
zjut001 12345678 1 1 计科04
```

4.2.12 修改账号状态

管理员可以修改账号状态（例如：解除账号锁定）。

```
请输入您要修改教师账号状态的用户名:
zjut001
请输入您要修改的状态(1:活跃 2:锁定):1
修改成功!
```

可以看到数据已经更新到 Manager.txt 文件中了。

```
ZJUT 123456 2 1 0
zjut003 123456 1 1 网工01
zjut002 123456 1 1 软工01
zjut001 12345678 1 2 计科04
```

```
zjut001 12345678 1 1 计科04
zjut002 123456 1 1 软工01
zjut003 123456 1 1 网工01
ZJUT 123456 2 1 0
```

4.3 选择 3 退出系统

```
▯=======================
|感谢您使用学生成绩管理系统~ |
=========================
```

4.4 细节展示

全系统的菜单均可返回上一级，即整个系统的菜单是连续连贯的。输入错误后随时可以返回上一级进行重新输入。

```
▯=========Menu==========
|欢迎进入学生成绩管理系统!   |
|1.教师登录               |
|2.管理员登陆             |
|3.退出系统               |
=========================
请输入:1
▯请输入您的身份(1:教师 2:管理员):
1
请输入账号:
zjut001
请输入密码:
12345678
▯======教师功能菜单=======
|1. 添加学生信息          |
|2. 查询学生信息          |
|3. 排序学生成绩          |
|4. 统计学生成绩          |
|5. 修改学生信息          |
|6. 删除学生信息          |
|7. 返回上级菜单          |
=========================
请输入:7
▯=========Menu==========
|欢迎进入学生成绩管理系统!   |
|1.教师登录               |
|2.管理员登陆             |
|3.退出系统               |
=========================
请输入:3
▯=======================
|感谢您使用学生成绩管理系统~ |
=========================
```

对所有数据都有正确性校验，如：添加已经存在的学生会输出错误信息。

```
请输入您要添加学生的姓名:
zxy
请输入您要添加学生的性别:
女
请输入您要添加学生的学号:
302023315168
请输入您要添加学生的班级:
计科04
请输入您要添加学生的成绩(四门分别是高数,程C,离散,大物.请按顺序填写!):
100 100 100 100
您添加的学生已经存在!
```

遇到的问题及解决方法如下：

- **问题 1：**

问题描述：教师在进行排序等操作时，代码会新建一个链表，将本班级的学生信息放入其中，完成操作后释放内存。但由于管理员在进行相同操作时，代码在原链表上进行操作，导致释放内存时把原链表直接释放了。

解决方法：一个方法是，用 const 修饰，并删除管理员模块中释放内存的代码；另一个方法是，拷贝一个新链表，在释放内存时释放新链表。为了保持代码的一致性，我选择了第二种方法。在 Stulist 类中添加了拷贝构造函数。

```
Stulist(const Stulist &other) {
    head = new Student;
    head->next = nullptr;
    size = 0;

    Student *p = other.head->next;
    Student *tail = head;
    while (p) {
        Student *newNode = new Student(*p);

        tail->next = newNode;
        tail = newNode;

        p = p->next;
    }
}
```

- **问题 2：**

问题描述：在 main 程序中，主要通过 while、switch case、goto 方法来进行界面的跳

转工作。但是在编译时一直报错。在经过查询后发现了问题所在：由于变量作用域的不确定性，编译出错。

解决方法：在每一个 case 后加一个中括号即可解决。

五、 实验总结（优点、不足、收获及体会）

我设计的学生成绩管理系统满足了全部任务书的功能要求，类的结构和关系清晰，功能完善。在任务书的基础上，细化为教师模块和管理员模块。与此同时，我还增加了许多我认为有必要的限制，比如：教师和管理员权限的分配；账号密码输入错误的锁定机制；在修改学生信息时不允许修改学号等等。

存在的优点：1. 类的结构和关系清晰，功能完善。细化教师模块和管理员模块的功能以及权限分配 2.实现了模糊查询的功能，可以根据姓名、班级进行模糊查询。3.高封装度和代码耦合度，将菜单 Menu 都封装成类，主程序仅保留必要的逻辑，使得代码耦合度高封装度高，便于快速阅读和理解代码。

存在的缺点：采用单向链表，在查找上没有优势，查找时间复杂度较高，希望在今后的学习中能找出高效的查找方法；虽然对一些通用的函数进行了封装，但代码仍存在冗余的问题；功能延伸较少，虽然系统功能完善，但还有很大的提升空间。

通过这次 C++的大型实验，我深刻的明白到：课本知识与实践能力相结合的重要性。只知道看书学书本上的东西，只不过是“纸上谈兵”，根本没有办法处理这些繁琐复杂的问题。一定要自己写，修 bug，从 bug 中汲取错误的经验教训，提升自己的代码能力。

在编程的过程中，要培养自己遵循良好的编程习惯。例如，避免重复的代码，注重代码的可读性和可维护性，合理选择变量名等等。已经有了一些项目经验，我更明白代码封装，注释，逻辑以及团队协作的重要性。作为计算机专业的学生，一定要从一开始就培养好良好的编程习惯，变量名规范化，合理的写注释。

最后也要感谢毛老师的谆谆教诲，是老师的正确指导和细心教学，我才能掌握这些繁复的 C++知识，并最终独立完成这门课程的课程设计。

六、 附录：源代码

附上 github 地址

qianqianzyk/c-programme (github.com)

main.cpp

```
1.  #include <iostream>
2.  #include <fstream>
3.  #include "./hpp/Menu.hpp"
4.  #include "./hpp/Manlist.hpp"
5.  #include "./hpp/Stulist.hpp"
```

```
6.
7.  using namespace std;
8.
9.  int main() {
10.     Stulist stulist;
11.     Manlist manlist;
12.
13.     ifstream inStudent("../txt/Student.txt");
14.     ifstream inManager("../txt/Manager.txt");
15.
16.     // 读取学生信息
17.     string name, gender, id, className;
18.     int scores[4];
19.     while (inStudent >> name >> gender >> id >> className) {
20.         for (int i = 0; i < 4; ++i) inStudent >> scores[i];
21.         stulist.addStudent(name, gender, id, className, scores);
22.     }
23.     inStudent.close();
24.
25.     // 读取管理员信息
26.     string username, password, classmanage;
27.     int userType, status;
28.     while (inManager >> username >> password >> userType >> status >>
     classmanage) {
29.         manlist.addManager(username, password, userType, status, clas
    smanage);
30.     }
31.     inManager.close();
32.
33.     int choice_1, choice_2, choice_3, choice_4, choice_5, choice_6;
34.
35.     process1:
36.     Menu::welcome();
37.     while (cin >> choice_1) {
38.         switch (choice_1) {
39.             case 1: {
40.                 Manager *target = manlist.login();
41.                 if (target == nullptr) goto process1;
42.                 process2:
43.                 Menu::teacherMenu();
44.                 cin >> choice_2;
45.                 switch (choice_2) {
46.                     case 1: {//添加学生信息
```

```
47.                         stulist.addStudentByTeacher(target->getClassM
    anage());
48.                         goto process2;
49.                         break;
50.                     }
51.                     case 2: {//查询学生信息
52.                         Menu::howFind();
53.                         cin >> choice_3;
54.                         switch (choice_3) {
55.                             case 1: {//根据学号
56.                                 stulist.findStudentByID(target->getCl
    assManage());
57.                                 goto process2;
58.                                 break;
59.                             }
60.                             case 2: {//根据姓名
61.                                 stulist.findStudentsByName(target->ge
    tClassManage());
62.                                 goto process2;
63.                                 break;
64.                             }
65.                             case 3: {//根据班级
66.                                 stulist.findStudentsByClass(target->g
    etClassManage());
67.                                 goto process2;
68.                                 break;
69.                             }
70.                             case 4: {
71.                                 goto process2;
72.                                 break;
73.                             }
74.                             default: {
75.                                 cout << "请输入有效值!" << endl;
76.                                 goto process2;
77.                                 break;
78.                             }
79.
80.                         }
81.                     }
82.                     case 3: {//排序
83.                         Menu::howSort();
84.                         cin >> choice_4;
85.                         switch (choice_4) {
86.                             case 1: {//根据学号
```

```
87.                                 stulist.sortStudentsByID(target->getC
    lassManage());
88.                                 goto process2;
89.                                 break;
90.                             }
91.                             case 2: {//根据单科成绩
92.                                 stulist.sortStudentsBySubjectScore(ta
    rget->getClassManage());
93.                                 goto process2;
94.                                 break;
95.                             }
96.                             case 3: {//根据平均分
97.                                 stulist.sortStudentsByAverageScore(ta
    rget->getClassManage());
98.                                 goto process2;
99.                                 break;
100.                            }
101.                            case 4: {//根据总分
102.                                stulist.sortStudentsByTotalScore(t
    arget->getClassManage());
103.                                goto process2;
104.                                break;
105.                            }
106.                            case 5: {
107.                                goto process2;
108.                                break;
109.                            }
110.                            default: {
111.                                cout << "请输入有效值!" << endl;
112.                                goto process2;
113.                                break;
114.                            }
115.                        }
116.                    }
117.                    case 4: {//统计学生成绩
118.                        stulist.countStudentsScore(target->getClas
    sManage());
119.                        goto process2;
120.                        break;
121.                    }
122.                    case 5: {//修改学生信息
123.                        stulist.updateStudentByID(target->getClass
    Manage());
124.                        goto process2;
```

```
125.                         break;
126.                     }
127.                     case 6: {//删除学生信息
128.                         stulist.deleteStudentByID(target->getClass
    Manage());
129.                         goto process2;
130.                         break;
131.                     }
132.                     case 7: {
133.                         goto process1;
134.                         break;
135.                     }
136.                     default: {
137.                         cout << "请输入有效值!" << endl;
138.                         goto process2;
139.                         break;
140.                     }
141.                 }
142.                 break;
143.             }
144.             case 2: {
145.                 Manager *target = manlist.login();
146.                 if (target == nullptr) goto process1;
147.                 process3:
148.                 Menu::managerMenu();
149.                 cin >> choice_5;
150.                 switch (choice_5) {
151.                     case 1: {//管理员添加学生信息
152.                         manlist.addStudentByManager(stulist);
153.                         goto process3;
154.                         break;
155.                     }
156.                     case 2: {//查询学生信息
157.                         Menu::howFind();
158.                         cin >> choice_5;
159.                         switch (choice_5) {
160.                             case 1: {//根据学号
161.                                 manlist.managerFindStudentByID(stu
    list);
162.                                 goto process3;
163.                                 break;
164.                             }
165.                             case 2: {//根据姓名
```

```
166.                                manlist.managerFindStudentsByName(
    stulist);
167.                                goto process3;
168.                                break;
169.                            }
170.                            case 3: {//根据班级
171.                                manlist.managerFindStudentsByClass
    (stulist);
172.                                goto process3;
173.                                break;
174.                            }
175.                            case 4: {
176.                                goto process3;
177.                                break;
178.                            }
179.                            default: {
180.                                cout << "请输入有效值!" << endl;
181.                                goto process3;
182.                                break;
183.                            }
184.                        }
185.                    }
186.                    case 3: {//排序
187.                        Menu::howSort();
188.                        cin >> choice_6;
189.                        switch (choice_6) {
190.                            case 1: {//根据学号
191.                                manlist.managerSortStudentsByID(st
    ulist);
192.                                goto process3;
193.                                break;
194.                            }
195.                            case 2: {//根据单科成绩
196.                                manlist.managerSortStudentsBySubje
    ctScore(stulist);
197.                                goto process3;
198.                                break;
199.                            }
200.                            case 3: {//根据平均分
201.                                manlist.managerSortStudentsByAvera
    geScore(stulist);
202.                                goto process3;
203.                                break;
204.                            }
```

```
205.                                case 4: {//根据总分
206.                                    manlist.managerSortStudentsByTotal
    Score(stulist);
207.                                    goto process3;
208.                                    break;
209.                                }
210.                                case 5: {
211.                                    goto process3;
212.                                    break;
213.                                }
214.                                default: {
215.                                    cout << "请输入有效值!" << endl;
216.                                    goto process3;
217.                                    break;
218.                                }
219.                            }
220.                        }
221.                        case 4: {//统计
222.                            manlist.managerCountStudentsScore(stulist)
    ;
223.                            goto process3;
224.                            break;
225.                        }
226.                        case 5: {//修改学生信息
227.                            manlist.managerUpdateStudentByID(stulist);

228.                            goto process3;
229.                            break;
230.                        }
231.                        case 6: {//删除学生信息
232.                            manlist.managerDeleteStudentByID(stulist);

233.                            goto process3;
234.                            break;
235.                        }
236.                        case 7: {//添加教师账号
237.                            manlist.addTeacherByManager();
238.                            goto process3;
239.                            break;
240.                        }
241.                        case 8: {//删除教师账号
242.                            manlist.delManager();
243.                            goto process3;
244.                            break;
```

```
                    }
                    case 9: {//重置账号密码
                        manlist.resetPassword();
                        goto process3;
                        break;
                    }
                    case 10: {//接触账号锁定
                        manlist.setAccount();
                        goto process3;
                        break;
                    }
                    case 11: {
                        goto process1;
                        break;
                    }
                    default: {
                        cout << "请输入有效值!" << endl;
                        goto process3;
                        break;
                    }
                }
                break;
            }
            case 3: {
                Menu::bye();
                return 0;
                break;
            }
            default: {
                cout << "请输入有效值!" << endl;
                goto process1;
                break;
            }
        }
    }
    return 0;
}
```

Menu.hpp

```
#ifndef C_PROGRAMME_MENU_H
#define C_PROGRAMME_MENU_H

#include <iostream>

using namespace std;
```

```
7.
8.  class Menu {
9.  public:
10.     static void welcome();
11.
12.     static void teacherMenu();
13.
14.     static void managerMenu();
15.
16.     static void bye();
17.
18.     static void howFind();
19.
20.     static void howCount();
21.
22.     static void howSort();
23.
24.     static void countScore();
25.
26. private:
27.     static void clearScreen() {
28.         system("cls");
29.     }
30. };
31.
32. #endif
```

Menu.cpp

```
1.  #include "../hpp/Menu.hpp"
2.
3.  using namespace std;
4.
5.  void Menu::welcome() {
6.      clearScreen();
7.      cout << "=========Menu==========" << endl;
8.      cout << "|欢迎进入学生成绩管理系统!   |" << endl;
9.      cout << "|1.教师登录               |" << endl;
10.     cout << "|2.管理员登陆             |" << endl;
11.     cout << "|3.退出系统               |" << endl;
12.     cout << "=======================" << endl;
13.     cout << "请输入:";
14. }
15.
16. void Menu::teacherMenu() {
17.     clearScreen();
```

```
18.     cout << "======教师功能菜单=======" << endl;
19.     cout << "|1. 添加学生信息          |" << endl;
20.     cout << "|2. 查询学生信息          |" << endl;
21.     cout << "|3. 排序学生成绩          |" << endl;
22.     cout << "|4. 统计学生成绩          |" << endl;
23.     cout << "|5. 修改学生信息          |" << endl;
24.     cout << "|6. 删除学生信息          |" << endl;
25.     cout << "|7. 返回上级菜单          |" << endl;
26.     cout << "========================" << endl;
27.     cout << "请输入:";
28. }
29.
30. void Menu::managerMenu() {
31.     clearScreen();
32.     cout << "=====管理员功能菜单=====" << endl;
33.     cout << "|1.  添加学生信息         |" << endl;
34.     cout << "|2.  查询学生信息         |" << endl;
35.     cout << "|3.  排序学生成绩         |" << endl;
36.     cout << "|4.  统计学生成绩         |" << endl;
37.     cout << "|5.  修改学生信息         |" << endl;
38.     cout << "|6.  删除学生信息         |" << endl;
39.     cout << "|7.  添加教师账号         |" << endl;
40.     cout << "|8.  删除教师账号         |" << endl;
41.     cout << "|9.  重置账号密码         |" << endl;
42.     cout << "|10. 修改账号状态         |" << endl;
43.     cout << "|11. 返回上级菜单         |" << endl;
44.     cout << "========================" << endl;
45.     cout << "请输入:";
46. }
47.
48. void Menu::bye() {
49.     clearScreen();
50.     cout << "========================" << endl;
51.     cout << "|感谢您使用学生成绩管理系统~ |" << endl;
52.     cout << "========================" << endl;
53. }
54.
55. void Menu::howFind() {
56.     clearScreen();
57.     cout << "========================" << endl;
58.     cout << "|1. 根据学生学号          |" << endl;
59.     cout << "|2. 根据学生姓名          |" << endl;
60.     cout << "|3. 根据学生班级          |" << endl;
61.     cout << "|4. 返回上级菜单          |" << endl;
```

```
62.     cout << "======================" << endl;
63.     cout << "请输入:";
64. }
65.
66. void Menu::howSort() {
67.     clearScreen();
68.     cout << "======================" << endl;
69.     cout << "|1. 根据学生学号          |" << endl;
70.     cout << "|2. 根据单科成绩          |" << endl;
71.     cout << "|3. 根据平均成绩          |" << endl;
72.     cout << "|4. 根据学生总分          |" << endl;
73.     cout << "|5. 返回上级菜单          |" << endl;
74.     cout << "======================" << endl;
75.     cout << "请输入:";
76. }
```

Student.hpp

```
1.  #ifndef C_PROGRAMME_STUDENT_H
2.  #define C_PROGRAMME_STUDENT_H
3.
4.  #include <iostream>
5.  #include <cstring>
6.
7.  using namespace std;
8.
9.  class Student {
10. private:
11.     string name;
12.     string gender;
13.     string id;
14.     string className;
15.     int scores[4];
16.     int totalScore;
17.     double averageScore;
18. public:
19.     Student *next;
20.
21.     // 构造函数
22.     Student(string namev = "", string genderv = "", string idv = "",
    string classNamev = "")
23.             : name(namev), gender(genderv), id(idv), className(classN
    amev), totalScore(0), averageScore(0.0) {
24.         for (int i = 0; i < 4; ++i) {
25.             scores[i] = 0;
26.         }
```

```
27.     }
28.
29.     string getName() const { return name; }
30.
31.     string getGender() const { return gender; }
32.
33.     string getId() const { return id; }
34.
35.     string getClassName() const { return className; }
36.
37.     const int *getScores() const { return scores; }
38.
39.     int getTotalScore() const { return totalScore; }
40.
41.     double getAverageScore() const { return averageScore; }
42.
43.     void setName(const string &namev) { name = namev; }
44.
45.     void setGender(const string &genderv) { gender = genderv; }
46.
47.     void setId(const string &idv) { id = idv; }
48.
49.     void setClassName(const string &classNamev) { className = classNa
    mev; }
50.
51.     void setScores(const int (&scoresv)[4]) { memcpy(scores, scoresv,
     sizeof(scores)); }
52.
53.     // 计算总分
54.     void calculateTotalScore();
55.
56.     // 计算个人平均分
57.     void calculateAverageScore();
58.
59.     // 初始化学生成绩
60.     void readScores(int (&scores)[4]);
61. };
62.
63.
64. #endif
```

Student.cpp

```
1.  #include "../hpp/Student.hpp"
2.
3.  using namespace std;
```

```
4.
5.  void Student::calculateTotalScore() {
6.      totalScore = 0;
7.      for (int i = 0; i < 4; ++i) {
8.          totalScore += scores[i];
9.      }
10. }
11.
12. void Student::calculateAverageScore() {
13.     averageScore = totalScore / 4.0;
14. }
15.
16. void Student::readScores(int (&scoresv)[4]) {
17.     for (int i = 0; i < 4; ++i) {
18.         this->scores[i] = scoresv[i];
19.     }
20.     calculateTotalScore();
21.     calculateAverageScore();
22. }
```

Stulist.hpp

```
1.  #ifndef C_PROGRAMME_STULIST_H
2.  #define C_PROGRAMME_STULIST_H
3.
4.  #include <iostream>
5.  #include "../hpp/Student.hpp"
6.
7.  using namespace std;
8.
9.  class Stulist {
10. private:
11.     Student *head;
12.     int size;
13. public:
14.     Stulist() {
15.         head = new Student;
16.         head->next = nullptr;
17.         size = 0;
18.     }
19.
20.     Stulist(const Stulist &other) {
21.         head = new Student;
22.         head->next = nullptr;
23.         size = 0;
24.
```

```
25.         Student *p = other.head->next;
26.         Student *tail = head;
27.         while (p) {
28.             Student *newNode = new Student(*p);
29.
30.             tail->next = newNode;
31.             tail = newNode;
32.
33.             p = p->next;
34.         }
35.     }
36.
37.     // 获取表头
38.     Student *getHead() { return head; }
39.
40.     // 添加学生
41.     void addStudent(string &namev, string &genderv, string &idv, stri
    ng &classNamev, int (&scoresv)[4]);
42.
43.     // 老师添加学生
44.     void addStudentByTeacher(string classmanagev);
45.
46.     // 查看学生是否已经存在
47.     Student *checkIfExist(string idv);
48.
49.     // 写入文件
50.     void write();
51.
52.     // 根据学号查找学生
53.     void findStudentByID(string classmanagev);
54.
55.     // 根据姓名查找学生(模糊查找)
56.     void findStudentsByName(string classmanagev);
57.
58.     // 根据班级查找学生
59.     void findStudentsByClass(string classmanagev);
60.
61.     // 根据学号升序排序
62.     void sortStudentsByID(string classmanagev);
63.
64.     // 根据单科成绩降序排序
65.     void sortStudentsBySubjectScore(string classmanagev);
66.
67.     // 根据总分降序排序
```

```
68.     void sortStudentsByTotalScore(string classmanagev);
69.
70.     // 根据平均分降序排序
71.     void sortStudentsByAverageScore(string classmanagev);
72.
73.     // 统计学生信息
74.     void countStudentsScore(string classmanagev);
75.
76.     // 查找同一个班的学生
77.     Student *findTheSameClassStudents(std::string classmanagev);
78.
79.     // 修改学生信息
80.     void updateStudentByID(string classmanagev);
81.
82.     // 删除学生信息
83.     void deleteStudentByID(string classmanagev);
84.
85.     // 展示所有学生
86.     void showAllStudents();
87.
88. };
89.
90. #endif
```

Stulist.cpp

```
1.  #include <iostream>
2.  #include <cstring>
3.  #include <fstream>
4.  #include <cctype>
5.  #include <algorithm>
6.  #include "../hpp/Stulist.hpp"
7.  #include "../hpp/Manlist.hpp"
8.
9.  using namespace std;
10.
11. void Stulist::addStudent(string &namev, string &genderv, string &idv,
     string &classNamev, int (&scoresv)[4]) {
12.     Student *p = new Student(namev, genderv, idv, classNamev);
13.     p->readScores(scoresv);
14.     p->next = head->next;
15.     head->next = p;
16.     size++;
17. }
18.
19. void Stulist::addStudentByTeacher(string classmanagev) {
```

```
20.     string namev, genderv, idv, classNamev;
21.     int scoresv[4];
22.     cout << "请输入您要添加学生的姓名:" << endl;
23.     cin >> namev;
24.     cout << "请输入您要添加学生的性别:" << endl;
25.     cin >> genderv;
26.     if (!(genderv == "男" || genderv == "女")){
27.         cout << "性别输入无效!" << endl;
28.         return;
29.     }
30.     cout << "请输入您要添加学生的学号:" << endl;
31.     cin >> idv;
32.     if (!(idv.length() == 12 && all_of(idv.begin(), idv.end(), ::isdi
    git))){
33.         cout << "学号输入无效!请重新输入(12 位数字)." << endl;
34.         return;
35.     }
36.     cout << "请输入您要添加学生的班级:" << endl;
37.     cin >> classNamev;
38.     cout << "请输入您要添加学生的成绩(四门分别是高数,程 C,离散,大物.请按顺
    序填写!):" << endl;
39.     for (int i = 0; i < 4; ++i) cin >> scoresv[i];
40.     if (classNamev == classmanagev) {
41.         Student *stu = checkIfExist(idv);
42.         if (stu != nullptr) {
43.             cout << "您添加的学生已经存在!" << endl;
44.             return;
45.         }
46.         Student *p = new Student(namev, genderv, idv, classNamev);
47.         p->readScores(scoresv);
48.         p->next = head->next;
49.         head->next = p;
50.         size++;
51.         write();
52.         cout << "添加成功!" << endl;
53.         cout << "姓名:" << namev << endl;
54.         cout << "性别:" << genderv << endl;
55.         cout << "学号:" << idv << endl;
56.         cout << "班级:" << classNamev << endl;
57.         cout << "成绩(高数,程 C,离散,大物):";
58.         for (int i = 0; i < 4; ++i) {
59.             cout << scoresv[i] << ' ';
60.         }
61.         cout << endl;
```

```
62.         cout << "----------------------------------" << endl;
63.     } else {
64.         cout << "很抱歉,您无权操作!请联系管理员." << endl;
65.     }
66. }
67.
68. Student *Stulist::checkIfExist(string idv) {
69.     Student *p = head->next;
70.     while (p) {
71.         if (p->getId() == idv)
72.             return p;
73.         p = p->next;
74.     }
75.     return nullptr;
76. }
77.
78. void Stulist::write() {
79.     ofstream out("../txt/Student.txt");
80.     Student *p = head->next;
81.     while (p) {
82.         out << p->getName() << ' ';
83.         out << p->getGender() << ' ';
84.         out << p->getId() << ' ';
85.         out << p->getClassName() << ' ' << endl;
86.         const int *scores = p->getScores();
87.         for (int i = 0; i < 4; ++i) {
88.             out << scores[i] << ' ';
89.         }
90.         out << endl;
91.         p = p->next;
92.     }
93.     out.close();
94. }
95.
96. void Stulist::findStudentByID(string classmanagev) {
97.     string idv;
98.     cout << "请输入您要查找学生的学号:" << endl;
99.     cin >> idv;
100.       Student *stu = checkIfExist(idv);
101.       if (stu == nullptr) {
102.           cout << "很抱歉,并未找到该学生!" << endl;
103.           return;
104.       } else {
105.           if (stu->getClassName() == classmanagev) {
```

```
106.                cout << "----------------------------------
    " << endl;
107.                cout << "查询成功!学生信息为:" << endl;
108.                cout << "姓名:" << stu->getName() << endl;
109.                cout << "性别:" << stu->getGender() << endl;
110.                cout << "学号:" << stu->getId() << endl;
111.                cout << "班级:" << stu->getClassName() << endl;
112.                cout << "成绩(高数,程C,离散,大物):";
113.                const int *scores = stu->getScores();
114.                for (int i = 0; i < 4; ++i) {
115.                    cout << scores[i] << ' ';
116.                }
117.                cout << endl;
118.                cout << "总分:" << stu->getTotalScore() << endl;
119.                cout << "平均分:" << stu->getAverageScore() << endl;
120.                cout << "----------------------------------
    " << endl;
121.            } else {
122.                cout << "很抱歉,您无权操作!请联系管理员." << endl;
123.            }
124.        }
125.    }
126.
127.    void Stulist::findStudentsByName(string classmanagev) {
128.        string namev;
129.        cout << "请输入您要查找学生的姓名(支持模糊查找):" << endl;
130.        cin >> namev;
131.
132.        bool found = false;
133.        Student *p = head->next;
134.        while (p) {
135.            if (p->getName().find(namev) != string::npos && p->getClas
    sName() == classmanagev) {
136.                found = true;
137.                cout << "姓名:" << p->getName() << endl;
138.                cout << "性别:" << p->getGender() << endl;
139.                cout << "学号:" << p->getId() << endl;
140.                cout << "班级:" << p->getClassName() << endl;
141.                cout << "成绩(高数,程C,离散,大物):";
142.                const int *scores = p->getScores();
143.                for (int i = 0; i < 4; ++i) {
144.                    cout << scores[i] << ' ';
145.                }
146.                cout << endl;
```

```
147.                cout << "总分:" << p->getTotalScore() << endl;
148.                cout << "平均分:" << p->getAverageScore() << endl;
149.                cout << "----------------------------------
    " << endl;
150.            }
151.            p = p->next;
152.        }
153.        if (!found) {
154.            cout << "未找到符合条件的学生.请确认是否是您班级里的学
    生!" << endl;
155.            return;
156.        } else {
157.            cout << "查询成功!找到以上学生." << endl;
158.            return;
159.        }
160.    }
161.
162.    void Stulist::findStudentsByClass(string classmanagev) {
163.        cout << "请注意,查询到的为您班级上的学生.如果未查到指定的学生,可能是
    因为该学生为其他班级上的或者该学生的信息并未填写提交!"
164.             << endl;
165.        cout << "----------------------------------------------------
    ----------------------------------------" << endl;
166.        cout << "查询成功!找到以下学生." << endl;
167.        cout << "如果想要获取学生的详细信息,请使用学号或姓名查
    询!" << endl;
168.
169.        Student *p = head->next;
170.        while (p) {
171.            if (p->getClassName() == classmanagev) {
172.                cout << "姓名          学号" << endl;
173.                cout << p->getName() << "          " << p->getId() << e
    ndl;
174.            }
175.            p = p->next;
176.        }
177.        cout << "----------------------------------------------------
    ----------------------------------------" << endl;
178.    }
179.
180.    void Stulist::sortStudentsByID(string classmanagev) {
181.
182.        Student *classStudents = findTheSameClassStudents(classmanagev
    );
```

```
183.
184.       // 插入排序对链表中的学生按学号升序排序
185.       Student *sortedList = nullptr;
186.       while (classStudents) {
187.           // 从未排序链表中移除一个节点
188.           Student *current = classStudents;
189.           classStudents = classStudents->next;
190.           // 将节点插入到已排序链表中的正确位置
191.           if (!sortedList || current->getId() < sortedList->getId())
    {
192.               // 将节点插入到已排序链表的开头
193.               current->next = sortedList;
194.               sortedList = current;
195.           } else {
196.               // 在已排序链表中找到插入位置
197.               Student *temp = sortedList;
198.               while (temp->next && temp->next->getId() < current->ge
   tId()) {
199.                   temp = temp->next;
200.               }
201.               // 将节点插入到已排序链表的中间
202.               current->next = temp->next;
203.               temp->next = current;
204.           }
205.       }
206.
207.       cout << "排序成功!(按学号升序排序)" << endl << classmanagev << "
   班学生列表:" << endl;
208.       cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程 C||成绩.离散
   ||成绩.大物||总分||平均分)" << endl;
209.       cout << "-------------------------------------------------------
   ------------------------------" << endl;
210.       int number = 1;
211.       while (sortedList) {
212.           cout << number++ << " " << sortedList->getName() << " " <<
    sortedList->getGender() << " " << sortedList->getId()
213.                << " ";
214.           const int *scores = sortedList->getScores();
215.           for (int i = 0; i < 4; ++i) {
216.               cout << scores[i] << ' ';
217.           }
218.           cout << sortedList->getTotalScore() << " " << sortedList->
   getAverageScore() << endl;
219.           // 移动到下一个节点并释放当前节点的内存
```

```
220.            Student *temp = sortedList;
221.            sortedList = sortedList->next;
222.            delete temp;
223.        }
224.        cout << "-------------------------------------------------------
    ------------------------------" << endl;
225.    }
226.
227.    void Stulist::sortStudentsBySubjectScore(string classmanagev) {
228.
229.        Student *classStudents = findTheSameClassStudents(classmanagev
    );
230.
231.        int index;
232.        cout << "请输入您要根据哪门成绩进行排序(1:高数 2:程C 3:离散 4:大
    物 )" << endl;
233.        cin >> index;
234.        // 插入排序按指定科目成绩降序排序
235.        Student *sortedList = nullptr;
236.        while (classStudents) {
237.            // 从未排序链表中移除一个节点
238.            Student *current = classStudents;
239.            classStudents = classStudents->next;
240.            // 将节点插入到已排序链表中的正确位置
241.            if (!sortedList || current->getScores()[index - 1] > sorte
    dList->getScores()[index - 1]) {
242.                // 将节点插入到已排序链表的开头
243.                current->next = sortedList;
244.                sortedList = current;
245.            } else {
246.                // 在已排序链表中找到插入位置
247.                Student *temp = sortedList;
248.                while (temp->next && temp->next->getScores()[index - 1
    ] > current->getScores()[index - 1]) {
249.                    temp = temp->next;
250.                }
251.                // 将节点插入到已排序链表的中间
252.                current->next = temp->next;
253.                temp->next = current;
254.            }
255.        }
256.
257.        cout << "排序成功!(按单科成绩降序排
    序)" << endl << classmanagev << "班学生列表:" << endl;
```

```
258.       cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程 C||成绩.离散
   ||成绩.大物||总分||平均分)" << endl;
259.       cout << "-------------------------------------------------------
   ------------------------------" << endl;
260.       int number = 1;
261.       while (sortedList) {
262.           cout << number++ << " " << sortedList->getName() << " " <<
    sortedList->getGender() << " " << sortedList->getId()
263.                << " ";
264.           const int *scores = sortedList->getScores();
265.           for (int i = 0; i < 4; ++i) {
266.               cout << scores[i] << ' ';
267.           }
268.           cout << sortedList->getTotalScore() << " " << sortedList->
   getAverageScore() << endl;
269.           // 移动到下一个节点并释放当前节点的内存
270.           Student *temp = sortedList;
271.           sortedList = sortedList->next;
272.           delete temp;
273.       }
274.       cout << "-------------------------------------------------------
   ------------------------------" << endl;
275.   }
276.
277.   void Stulist::sortStudentsByTotalScore(string classmanagev) {
278.
279.       Student *classStudents = findTheSameClassStudents(classmanagev
   );
280.       // 插入排序按总分降序排序
281.       Student *sortedList = nullptr;
282.       while (classStudents) {
283.           // 从未排序链表中移除一个节点
284.           Student *current = classStudents;
285.           classStudents = classStudents->next;
286.           // 将节点插入到已排序链表中的正确位置
287.           if (!sortedList || current->getTotalScore() > sortedList->
   getTotalScore()) {
288.               // 将节点插入到已排序链表的开头
289.               current->next = sortedList;
290.               sortedList = current;
291.           } else {
292.               // 在已排序链表中找到插入位置
293.               Student *temp = sortedList;
```

```
294.                 while (temp->next && temp->next->getTotalScore() > cur
    rent->getTotalScore()) {
295.                     temp = temp->next;
296.                 }
297.                 // 将节点插入到已排序链表的中间
298.                 current->next = temp->next;
299.                 temp->next = current;
300.             }
301.         }
302.
303.         cout << "排序成功!(按总分降序排序)" << endl << classmanagev << "
    班学生列表:" << endl;
304.         cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程 C||成绩.离散
    ||成绩.大物||总分||平均分)" << endl;
305.         cout << "----------------------------------------------------
    ----------------------------" << endl;
306.         int number = 1;
307.         while (sortedList) {
308.             cout << number++ << " " << sortedList->getName() << " " <<
     sortedList->getGender() << " " << sortedList->getId()
309.                  << " ";
310.             const int *scores = sortedList->getScores();
311.             for (int i = 0; i < 4; ++i) {
312.                 cout << scores[i] << ' ';
313.             }
314.             cout << sortedList->getTotalScore() << " " << sortedList->
    getAverageScore() << endl;
315.             // 移动到下一个节点并释放当前节点的内存
316.             Student *temp = sortedList;
317.             sortedList = sortedList->next;
318.             delete temp;
319.         }
320.         cout << "----------------------------------------------------
    ----------------------------" << endl;
321.     }
322.
323.     void Stulist::sortStudentsByAverageScore(string classmanagev) {
324.
325.         Student *classStudents = findTheSameClassStudents(classmanagev
    );
326.
327.         // 插入排序按平均分降序排序
328.         Student *sortedList = nullptr;
329.         while (classStudents) {
```

```
330.            // 从未排序链表中移除一个节点
331.            Student *current = classStudents;
332.            classStudents = classStudents->next;
333.            // 将节点插入到已排序链表中的正确位置
334.            if (!sortedList || current->getAverageScore() > sortedList
    ->getAverageScore()) {
335.                // 将节点插入到已排序链表的开头
336.                current->next = sortedList;
337.                sortedList = current;
338.            } else {
339.                // 在已排序链表中找到插入位置
340.                Student *temp = sortedList;
341.                while (temp->next && temp->next->getAverageScore() > c
    urrent->getAverageScore()) {
342.                    temp = temp->next;
343.                }
344.                // 将节点插入到已排序链表的中间
345.                current->next = temp->next;
346.                temp->next = current;
347.            }
348.        }
349.
350.        cout << "排序成功!(按平均分降序排
    序)" << endl << classmanagev << "班学生列表:" << endl;
351.        cout << "(名次||姓名||性别||学号||成绩.高数||成绩.程 C||成绩.离散
    ||成绩.大物||总分||平均分)" << endl;
352.        cout << "-----------------------------------------------------
    ------------------------------" << endl;
353.        int number = 1;
354.        while (sortedList) {
355.            cout << number++ << " " << sortedList->getName() << " " <<
     sortedList->getGender() << " " << sortedList->getId()
356.                 << " ";
357.            const int *scores = sortedList->getScores();;
358.            for (int i = 0; i < 4; ++i) {
359.                cout << scores[i] << ' ';
360.            }
361.            cout << sortedList->getTotalScore() << " " << sortedList->
    getAverageScore() << endl;
362.            // 移动到下一个节点并释放当前节点的内存
363.            Student *temp = sortedList;
364.            sortedList = sortedList->next;
365.            delete temp;
366.        }
```

```
367.        cout << "-------------------------------------------------------
    ------------------------------" << endl;
368.    }
369.
370.    void Stulist::countStudentsScore(string classmanagev) {
371.
372.        Student *classStudents = findTheSameClassStudents(classmanagev
    );
373.
374.        // 插入排序按总分降序排序
375.        Student *sortedList = nullptr;
376.        while (classStudents) {
377.            // 从未排序链表中移除一个节点
378.            Student *current = classStudents;
379.            classStudents = classStudents->next;
380.            // 将节点插入到已排序链表中的正确位置
381.            if (!sortedList || current->getTotalScore() > sortedList->
    getTotalScore()) {
382.                // 将节点插入到已排序链表的开头
383.                current->next = sortedList;
384.                sortedList = current;
385.            } else {
386.                // 在已排序链表中找到插入位置
387.                Student *temp = sortedList;
388.                while (temp->next && temp->next->getTotalScore() > cur
    rent->getTotalScore()) {
389.                    temp = temp->next;
390.                }
391.                // 将节点插入到已排序链表的中间
392.                current->next = temp->next;
393.                temp->next = current;
394.            }
395.        }
396.
397.        cout << "统计成功!" << endl << classmanagev << "班学生成
    绩:" << endl;
398.        cout << "-------------------------------------------------------
    ------------------------------" << endl;
399.        int number = 1;
400.        double totalAverage = 0.0;
401.        int numAbove60[4] = {0};
402.        double totalSubjectScores[4] = {0};
403.        while (sortedList) {
404.            const int *scores = sortedList->getScores();
```

```
405.            for (int i = 0; i < 4; ++i) {
406.                totalSubjectScores[i] += scores[i];
407.                if (scores[i] > 60) {
408.                    numAbove60[i]++;
409.                }
410.            }
411.            totalAverage += sortedList->getTotalScore();
412.
413.            Student *temp = sortedList;
414.            sortedList = sortedList->next;
415.            delete temp;
416.            number++;
417.        }
418.        // 输出班级总人数
419.        cout << "班级总人数:" << number - 1 << endl;
420.        cout << endl;
421.        // 输出各科成绩平均分
422.        cout << "各科成绩平均分:" << endl;
423.        const char *subjectNames[4] = {"高数", "程 C", "离散", "大物
    "};
424.        for (int i = 0; i < 4; ++i) {
425.            cout << subjectNames[i] << ": " << totalSubjectScores[i] /
     ((number - 1) * 1.0) << endl;
426.        }
427.        cout << endl;
428.        // 输出各科成绩超过 60 分的人数
429.        cout << "各科及格的人数:" << endl;
430.        for (int i = 0; i < 4; ++i) {
431.            cout << subjectNames[i] << ": " << numAbove60[i] << endl;

432.        }
433.        cout << endl;
434.        // 输出总分的平均分
435.        cout << "班级平均
    分:" << totalAverage / ((number - 1) * 1.0) << endl;
436.        cout << "----------------------------------------------------
    ------------------------------" << endl;
437.    }
438.
439.    Student *Stulist::findTheSameClassStudents(string classmanagev) {

440.        // 创建一个新的链表来存储给定班级的学生
441.        Student *classStudents = nullptr;
442.        // 遍历链表，将给定班级的学生添加到新链表中
```

```
443.       Student *p = head->next;
444.       while (p) {
445.           if (p->getClassName() == classmanagev) {
446.               Student *newStudent = new Student(*p);
447.               newStudent->next = classStudents;
448.               classStudents = newStudent;
449.           }
450.           p = p->next;
451.       }
452.       return classStudents;
453.   }
454.
455.   void Stulist::updateStudentByID(string classmanagev) {
456.       string idv;
457.       cout << "请输入您要修改学生信息的学号:" << endl;
458.       cin >> idv;
459.       Student *stu = checkIfExist(idv);
460.       if (stu == nullptr) {
461.           cout << "很抱歉，并未找到该学生!" << endl;
462.           return;
463.       } else {
464.           if (stu->getClassName() == classmanagev) {
465.               cout << "----------------------------------
    " << endl;
466.               cout << "当前学生信息为:" << endl;
467.               cout << "姓名:" << stu->getName() << endl;
468.               cout << "性别:" << stu->getGender() << endl;
469.               cout << "学号:" << stu->getId() << endl;
470.               cout << "班级:" << stu->getClassName() << endl;
471.               cout << "成绩(高数,程C,离散,大物):";
472.               const int *scores = stu->getScores();
473.               for (int i = 0; i < 4; ++i) {
474.                   cout << scores[i] << ' ';
475.               }
476.               cout << endl;
477.               cout << "总分:" << stu->getTotalScore() << endl;
478.               cout << "平均分:" << stu->getAverageScore() << endl;
479.               cout << "----------------------------------
    " << endl;
480.
481.               cout << "请输入修改后的信息(注意:如果信息不变请输入原来
    的!):" << endl;
482.               string namev, genderv;
483.               int scoresv[4];
```

```
484.                cout << "姓名:" << endl;
485.                cin >> namev;
486.                cout << "性别:" << endl;
487.                cin >> genderv;
488.                if (!(genderv == "男" || genderv == "女")){
489.                    cout << "性别输入无效!" << endl;
490.                    return;
491.                }
492.                cout << "成绩(高数,程C,离散,大物):" << endl;
493.                for (int i = 0; i < 4; ++i) cin >> scoresv[i];
494.
495.                stu->setName(namev);
496.                stu->setGender(genderv);
497.                stu->readScores(scoresv);
498.
499.                write();
500.                cout << "学生信息修改成功!" << endl;
501.                cout << "姓名:" << stu->getName() << endl;
502.                cout << "性别:" << stu->getGender() << endl;
503.                cout << "学号:" << stu->getId() << endl;
504.                cout << "班级:" << stu->getClassName() << endl;
505.                cout << "成绩(高数,程C,离散,大物):";
506.                for (int i = 0; i < 4; ++i) {
507.                    cout << scores[i] << ' ';
508.                }
509.                cout << endl;
510.                cout << "----------------------------------
    " << endl;
511.            } else {
512.                cout << "很抱歉,您无权操作!请联系管理员." << endl;
513.            }
514.        }
515.    }
516.
517.    void Stulist::deleteStudentByID(string classmanagev) {
518.        string idv;
519.        cout << "请输入您要删除学生信息的学号:" << endl;
520.        cin >> idv;
521.
522.        Student *current = head->next;
523.        Student *pre = nullptr;
524.        while (current != nullptr) {
525.            if (current->getId() == idv && classmanagev != current->ge
    tClassName()) {
```

```
526.                 cout << "很抱歉,您无权操作!请联系管理员." << endl;
527.                 return;
528.             }
529.             if (current->getId() == idv && classmanagev == current->ge
    tClassName()) {
530.                 if (pre == nullptr) {
531.                     head->next = current->next;
532.                 } else {
533.                     pre->next = current->next;
534.                 }
535.                 delete current;
536.                 write();
537.                 cout << "成功删除学生信息!" << endl;
538.                 return;
539.             }
540.             pre = current;
541.             current = current->next;
542.         }
543.
544.         cout << "很抱歉，并未找到该学生!" << endl;
545.     }
546.
547.     void Stulist::showAllStudents() {
548.         Student *current = head->next;
549.         while (current != nullptr) {
550.             cout << "Name: " << current->getName() << endl;
551.             cout << "Gender: " << current->getGender() << endl;
552.             cout << "ID: " << current->getId() << endl;
553.             cout << "Class: " << current->getClassName() << endl;
554.             cout << "Scores:";
555.             const int *scores = current->getScores();
556.             for (int i = 0; i < 4; ++i) {
557.                 cout << scores[i] << ' ';
558.             }
559.             cout << endl;
560.             cout << "Total Score:" << current->getTotalScore() << endl
    ;
561.             cout << "Average Score:" << current->getAverageScore() <<
    endl;
562.             cout << "----------------------------------------
    " << endl;
563.
564.             current = current->next;
565.         }
```

```
566.   }
```

Manager.hpp

```
1.  #ifndef C_PROGRAMME_MANAGER_H
2.  #define C_PROGRAMME_MANAGER_H
3.
4.  #include <iostream>
5.  #include <cstring>
6.
7.  using namespace std;
8.
9.  class Manager {
10. public:
11.     Manager *next;
12.
13.     Manager(string usernamev = "", string passwordv = "", int userTyp
    ev = 1, int statusv = 1, string classmanagev = "")
14.             : username(usernamev),
15.               password(passwordv),
16.               userType(userTypev),
17.               status(statusv), classmanage(classmanagev) {};//构造函
    数
18.
19.     string getUsername() { return username; }//获取账号
20.
21.     string getPassword() { return password; }//获取密码
22.
23.     int getUserType() { return userType; }//获取用户类型
24.
25.     int getStatus() { return status; }//获取账号状态
26.
27.     string getClassManage() { return classmanage; }//获取权限
28.
29.     void setStatus(int s) { status = s; }//修改账号状态
30.
31.     void setPassword(string passwordv) { password = passwordv; }
32.
33. private:
34.     string username;//账号
35.     string password;//密码
36.     int userType; //用户类型(0:初始化 1:老师;2:管理员)
37.     int status; //账号状态(1:活跃;2:锁定)
38.     string classmanage; //管理的班级(管理员为 0;老师为对应的班级)
39. };
40.
```

```
41. #endif
```

Manager.cpp

```
1.  #include "../hpp/Manager.hpp"
```

Manlist.hpp

```
1.  #ifndef C_PROGRAMME_MANLIST_H
2.  #define C_PROGRAMME_MANLIST_H
3.
4.  #include "../hpp/Manager.hpp"
5.  #include "../hpp/Stulist.hpp"
6.
7.  using namespace std;
8.
9.  class Manlist {
10. private:
11.     Manager *head;
12.     int size;
13. public:
14.     Manlist() {
15.         head = new Manager;
16.         head->next = nullptr;
17.         size = 0;
18.     }
19.
20.     // 添加账号
21.     void addManager(string &usernamev, string &passwordv, int &userTy
    pev, int &statusv, string &classmanagev);
22.
23.     // 登录
24.     Manager *login();
25.
26.     // 管理员添加教师账号
27.     void addTeacherByManager();
28.
29.     // 检查登录
30.     Manager *checkIfLogin(string usernamev, string passwordv);
31.
32.     // 检查账号是否存在
33.     Manager *checkIfExist(string usernamev);
34.
35.     // 查找账号是否存在
36.     Manager *findManagerByUsername(const string &username);
37.
38.     // 管理员添加学生
39.     void addStudentByManager(Stulist &stulistv);
```

```
40.
41.     // 管理员根据学号查询学生
42.     void managerFindStudentByID(Stulist &stulistv);
43.
44.     // 管理员根据姓名查询学生
45.     void managerFindStudentsByName(Stulist &stulistv);
46.
47.     // 管理员根据班级查询学生
48.     void managerFindStudentsByClass(Stulist &stulistv);
49.
50.     // 管理员根据学号升序排序学生
51.     void managerSortStudentsByID(Stulist &stulistv);
52.
53.     // 管理员根据单科成绩降序排序学生
54.     void managerSortStudentsBySubjectScore(Stulist &stulistv);
55.
56.     // 管理员根据总分降序排序学生
57.     void managerSortStudentsByTotalScore(Stulist &stulistv);
58.
59.     // 管理员根据个人平均分降序排序学生
60.     void managerSortStudentsByAverageScore(Stulist &stulistv);
61.
62.     // 管理员根据班级统计学生成绩
63.     void managerCountStudentsScore(Stulist &stulistv);
64.
65.     // 管理员修改学生信息
66.     void managerUpdateStudentByID(Stulist &stulistv);
67.
68.     // 管理员删除学生信息
69.     void managerDeleteStudentByID(Stulist &stulistv);
70.
71.     // 删除账号
72.     void delManager();
73.
74.     // 重置账号密码
75.     void resetPassword();
76.
77.     // 写入文件
78.     void write();
79.
80.     // 锁定账号
81.     bool lockAccount(const string &username);
82.
83.     // 修改账号状态
```

```
84.     void setAccount();
85.
86.     // 获取所有账号
87.     void showAllManagers();
88. };
89.
90.
91. #endif //C_PROGRAMME_MANLIST_H
```

Manlist.cpp

```
1.  #include "../hpp/Stulist.hpp"
2.  #include "../hpp/Manlist.hpp"
3.  #include <vector>
4.  #include <algorithm>
5.  #include <fstream>
6.  #include <map>
7.
8.  using namespace std;
9.
10. void Manlist::addManager(string &usernamev, string &passwordv, int &u
    serTypev, int &statusv, string &classmanagev) {
11.     Manager *p = new Manager(usernamev, passwordv, userTypev, statusv
    , classmanagev);
12.     p->next = head->next;
13.     head->next = p;
14.     size++;
15. }
16.
17. Manager *Manlist::login() {
18.     system("cls");
19.     string usernamev, passwordv;
20.     int userTypev;
21.     cout << "请输入您的身份(1:教师 2:管理员):" << endl;
22.     cin >> userTypev;
23.     cout << "请输入账号:" << endl;
24.     cin >> usernamev;
25.     cout << "请输入密码:" << endl;
26.     cin >> passwordv;
27.
28.     Manager *checkstatus = findManagerByUsername(usernamev);
29.     if (userTypev == 2 && checkstatus->getUserType() != 2) {
30.         cout << "很抱歉,您不是管理员!" << endl;
31.         return nullptr;
32.     }
33.     if (checkstatus != nullptr && checkstatus->getStatus() == 2) {
```

```
34.         cout << "您的账号已被锁定!请联系管理员." << endl;
35.         return nullptr;
36.     }
37.     int wrongtime = 0;
38.     Manager *target = nullptr;
39.
40.     while (wrongtime < 5) {
41.         target = checkIfLogin(usernamev, passwordv);
42.         if (target != nullptr) {
43.             return target;
44.         } else {
45.             wrongtime++;
46.             cout << "账号或密码错误! 请注意,您还有
    " << 5 - wrongtime << "次机会!" << endl;
47.             if (wrongtime == 5) {
48.                 if (lockAccount(usernamev)) {
49.                     cout << "您已达到最大尝试次数,帐号已被锁定.请联系管理
    员!" << endl;
50.                 } else {
51.                     cout << "账号不存在!请联系管理员,并进行注
    册." << endl;
52.                 }
53.                 return nullptr;
54.             }
55.             cout << "请输入账号:" << endl;
56.             cin >> usernamev;
57.             cout << "请输入密码:" << endl;
58.             cin >> passwordv;
59.         }
60.     }
61.     write();
62.     return nullptr;
63. }
64.
65. void Manlist::addTeacherByManager() {
66.     string usernamev, passwordv, classmanagev;
67.     cout << "请输入您要添加教师账号的用户名:" << endl;
68.     cin >> usernamev;
69.     cout << "请输入您要添加教师账号的密码:" << endl;
70.     cin >> passwordv;
71.     cout << "请输入您要添加教师账号的权限:" << endl;
72.     cin >> classmanagev;
73.     Manager *man = checkIfExist(usernamev);
74.     if (man != nullptr) {
```

```
75.         cout << "您添加的教师账号已经存在!" << endl;
76.         return;
77.     }
78.     Manager *p = new Manager(usernamev, passwordv, 1, 1, classmanagev
    );
79.     p->next = head->next;
80.     head->next = p;
81.     size++;
82.     write();
83.     cout << "添加成功!" << endl;
84. }
85.
86. Manager *Manlist::checkIfLogin(string usernamev, string passwordv) {

87.     Manager *p = head->next;
88.     while (p) {
89.         if (p->getUsername() == usernamev && p->getPassword() == pass
    wordv)
90.             return p;
91.         p = p->next;
92.     }
93.     return nullptr;
94. }
95.
96. Manager *Manlist::checkIfExist(string usernamev) {
97.     Manager *p = head->next;
98.     while (p) {
99.         if (p->getUsername() == usernamev)
100.             return p;
101.         p = p->next;
102.       }
103.       return nullptr;
104.   }
105.
106.   Manager *Manlist::findManagerByUsername(const string &usernamev) {

107.       Manager *current = head->next;
108.       while (current != nullptr) {
109.           if (current->getUsername() == usernamev) {
110.               return current;
111.           }
112.           current = current->next;
113.       }
114.       return nullptr;
```

```
115.  }
116.
117.  void Manlist::addStudentByManager(Stulist &stulistv) {
118.      string namev, genderv, idv, classNamev;
119.      int scoresv[4];
120.      cout << "请输入您要添加学生的姓名:" << endl;
121.      cin >> namev;
122.      cout << "请输入您要添加学生的性别:" << endl;
123.      cin >> genderv;
124.      if (!(genderv == "男" || genderv == "女")){
125.          cout << "性别输入无效!" << endl;
126.          return;
127.      }
128.      cout << "请输入您要添加学生的学号:" << endl;
129.      cin >> idv;
130.      if (!(idv.length() == 12 && all_of(idv.begin(), idv.end(), ::i
    sdigit))){
131.          cout << "学号输入无效!请重新输入(12 位数字)." << endl;
132.          return;
133.      }
134.      cout << "请输入您要添加学生的班级:" << endl;
135.      cin >> classNamev;
136.      cout << "请输入您要添加学生的成绩(四门分别是高数,程 C,离散,大物.请按
    顺序填写!):" << endl;
137.      for (int i = 0; i < 4; ++i) cin >> scoresv[i];
138.      Student *stu = stulistv.checkIfExist(idv);
139.      if (stu != nullptr) {
140.          cout << "您添加的学生已经存在!" << endl;
141.          return;
142.      }
143.      stulistv.addStudent(namev, genderv, idv, classNamev, scoresv);

144.      stulistv.write();
145.      cout << "添加成功!" << endl;
146.      cout << "姓名:" << namev << endl;
147.      cout << "性别:" << genderv << endl;
148.      cout << "学号:" << idv << endl;
149.      cout << "班级:" << classNamev << endl;
150.      cout << "成绩(高数,程 C,离散,大物):";
151.      for (int i = 0; i < 4; ++i) {
152.          cout << scoresv[i] << ' ';
153.      }
154.      cout << endl;
155.      cout << "---------------------------------" << endl;
```

```
156.  }
157.
158.  void Manlist::managerFindStudentByID(Stulist &stulistv) {
159.      string idv;
160.      cout << "请输入您要查找学生的学号:" << endl;
161.      cin >> idv;
162.      Student *stu = stulistv.checkIfExist(idv);
163.      if (stu == nullptr) {
164.          cout << "很抱歉,并未找到该学生!" << endl;
165.          return;
166.      }
167.      cout << "----------------------------------" << endl;
168.      cout << "查询成功!学生信息为:" << endl;
169.      cout << "姓名:" << stu->getName() << endl;
170.      cout << "性别:" << stu->getGender() << endl;
171.      cout << "学号:" << stu->getId() << endl;
172.      cout << "班级:" << stu->getClassName() << endl;
173.      cout << "成绩(高数,程C,离散,大物):";
174.      const int *scores = stu->getScores();
175.      for (int i = 0; i < 4; ++i) {
176.          cout << scores[i] << ' ';
177.      }
178.      cout << endl;
179.      cout << "总分:" << stu->getTotalScore() << endl;
180.      cout << "平均分:" << stu->getAverageScore() << endl;
181.      cout << "----------------------------------" << endl;
182.  }
183.
184.  void Manlist::managerFindStudentsByName(Stulist &stulistv) {
185.      string namev;
186.      cout << "请输入您要查找学生的姓名(支持模糊查找):" << endl;
187.      cin >> namev;
188.
189.      bool found = false;
190.      Student *p = stulistv.getHead()->next;
191.      while (p) {
192.          if (p->getName().find(namev) != string::npos) {
193.              found = true;
194.              cout << "姓名:" << p->getName() << endl;
195.              cout << "性别:" << p->getGender() << endl;
196.              cout << "学号:" << p->getId() << endl;
197.              cout << "班级:" << p->getClassName() << endl;
198.              cout << "成绩(高数,程C,离散,大物):";
199.              const int *scores = p->getScores();
```

```
200.                for (int i = 0; i < 4; ++i) {
201.                    cout << scores[i] << ' ';
202.                }
203.                cout << endl;
204.                cout << "总分:" << p->getTotalScore() << endl;
205.                cout << "平均分:" << p->getAverageScore() << endl;
206.                cout << "----------------------------------
    " << endl;
207.            }
208.            p = p->next;
209.        }
210.        if (!found) {
211.            cout << "未找到符合条件的学生." << endl;
212.            return;
213.        } else {
214.            cout << "查询成功!找到以上学生." << endl;
215.            return;
216.        }
217.    }
218.
219.    void Manlist::managerFindStudentsByClass(Stulist &stulistv) {
220.        string classv;
221.        cout << "请输入您要查找学生的班级(支持模糊查找):" << endl;
222.        cin >> classv;
223.
224.        vector<string> matchClasses;
225.
226.        Student *p = stulistv.getHead()->next;
227.        while (p) {
228.            if (p->getClassName().find(classv) != string::npos) {
229.                matchClasses.push_back(p->getClassName());
230.            }
231.            p = p->next;
232.        }
233.
234.        sort(matchClasses.begin(), matchClasses.end());
235.        matchClasses.erase(unique(matchClasses.begin(), matchClasses.e
    nd()), matchClasses.end());
236.
237.        if (matchClasses.empty()) {
238.            cout << "未找到符合条件的班级." << endl;
239.            return;
240.        }
241.        cout << "查询成功!找到以下班级: " << endl;
```

```
242.       cout << "如果想要获取学生的详细信息,请使用学号或姓名查
    询!" << endl;
243.       cout << "-----------------------------" << endl;
244.       for (const string &className: matchClasses) {
245.           cout << "班级: " << className << endl;
246.           cout << "姓名          学号" << endl;
247.           p = stulistv.getHead()->next;
248.           while (p) {
249.               if (p->getClassName() == className) {
250.                   cout << p->getName() << "         " << p->getId()
    << endl;
251.               }
252.               p = p->next;
253.           }
254.           cout << "-----------------------------" << endl;
255.       }
256.   }
257.
258.   void Manlist::managerSortStudentsByID(Stulist &stulistv) {
259.       Stulist *s = new Stulist(stulistv);
260.       Student *students = s->getHead()->next;
261.       // 插入排序对链表中的学生按学号升序排序
262.       Student *sortedList = nullptr;
263.       while (students) {
264.           // 从未排序链表中移除一个节点
265.           Student *current = students;
266.           students = students->next;
267.           // 将节点插入到已排序链表中的正确位置
268.           if (!sortedList || current->getId() < sortedList->getId())
     {
269.               // 将节点插入到已排序链表的开头
270.               current->next = sortedList;
271.               sortedList = current;
272.           } else {
273.               // 在已排序链表中找到插入位置
274.               Student *temp = sortedList;
275.               while (temp->next && temp->next->getId() < current->ge
    tId()) {
276.                   temp = temp->next;
277.               }
278.               // 将节点插入到已排序链表的中间
279.               current->next = temp->next;
280.               temp->next = current;
281.           }
```

```
282.       }
283.
284.       cout << "排序成功!(按学号升序排序)" << endl;
285.       cout << "(名次||姓名||性别||学号||班级||成绩.高数||成绩.程 C||成
    绩.离散||成绩.大物||总分||平均分)" << endl;
286.       cout << "----------------------------------------------------
    ----------------------------" << endl;
287.       int number = 1;
288.       while (sortedList) {
289.           cout << number++ << " " << sortedList->getName() << " " <<
     sortedList->getGender() << " " << sortedList->getId()
290.                << " " << sortedList->getClassName() << " ";
291.           const int *scores = sortedList->getScores();
292.           for (int i = 0; i < 4; ++i) {
293.               cout << scores[i] << ' ';
294.           }
295.           cout << sortedList->getTotalScore() << " " << sortedList->
    getAverageScore() << endl;
296.           // 移动到下一个节点并释放当前节点的内存
297.           Student *temp = sortedList;
298.           sortedList = sortedList->next;
299.           delete temp;
300.       }
301.       cout << "----------------------------------------------------
    ----------------------------" << endl;
302.   }
303.
304.   void Manlist::managerSortStudentsBySubjectScore(Stulist &stulistv)
     {
305.       Stulist *s = new Stulist(stulistv);
306.       Student *students = s->getHead()->next;
307.       int index;
308.       cout << "请输入您要根据哪门成绩进行排序(1:高数 2:程 C 3:离散 4:大
    物 )" << endl;
309.       cin >> index;
310.       // 插入排序按指定科目成绩降序排序
311.       Student *sortedList = nullptr;
312.       while (students) {
313.           // 从未排序链表中移除一个节点
314.           Student *current = students;
315.           students = students->next;
316.           // 将节点插入到已排序链表中的正确位置
317.           if (!sortedList || current->getScores()[index - 1] > sorte
    dList->getScores()[index - 1]) {
```

```
318.                 // 将节点插入到已排序链表的开头
319.                 current->next = sortedList;
320.                 sortedList = current;
321.             } else {
322.                 // 在已排序链表中找到插入位置
323.                 Student *temp = sortedList;
324.                 while (temp->next && temp->next->getScores()[index - 1
    ] > current->getScores()[index - 1]) {
325.                     temp = temp->next;
326.                 }
327.                 // 将节点插入到已排序链表的中间
328.                 current->next = temp->next;
329.                 temp->next = current;
330.             }
331.         }
332.
333.         cout << "排序成功!(按单科成绩降序排序)" << endl;
334.         cout << "(名次||姓名||性别||学号||班级||成绩.高数||成绩.程 C||成
    绩.离散||成绩.大物||总分||平均分)" << endl;
335.         cout << "----------------------------------------------------
    ------------------------------" << endl;
336.         int number = 1;
337.         while (sortedList) {
338.             cout << number++ << " " << sortedList->getName() << " " <<
     sortedList->getGender() << " " << sortedList->getId()
339.                  << " " << sortedList->getClassName() << " ";
340.             const int *scores = sortedList->getScores();
341.             for (int i = 0; i < 4; ++i) {
342.                 cout << scores[i] << ' ';
343.             }
344.             cout << sortedList->getTotalScore() << " " << sortedList->
    getAverageScore() << endl;
345.             // 移动到下一个节点并释放当前节点的内存
346.             Student *temp = sortedList;
347.             sortedList = sortedList->next;
348.             delete temp;
349.         }
350.         cout << "----------------------------------------------------
    ------------------------------" << endl;
351.   }
352.
353.   void Manlist::managerSortStudentsByTotalScore(Stulist &stulistv) {

354.         Stulist *s = new Stulist(stulistv);
```

```
355.        Student *students = s->getHead()->next;
356.        // 插入排序按总分降序排序
357.        Student *sortedList = nullptr;
358.        while (students) {
359.            // 从未排序链表中移除一个节点
360.            Student *current = students;
361.            students = students->next;
362.            // 将节点插入到已排序链表中的正确位置
363.            if (!sortedList || current->getTotalScore() > sortedList->
    getTotalScore()) {
364.                // 将节点插入到已排序链表的开头
365.                current->next = sortedList;
366.                sortedList = current;
367.            } else {
368.                // 在已排序链表中找到插入位置
369.                Student *temp = sortedList;
370.                while (temp->next && temp->next->getTotalScore() > cur
    rent->getTotalScore()) {
371.                    temp = temp->next;
372.                }
373.                // 将节点插入到已排序链表的中间
374.                current->next = temp->next;
375.                temp->next = current;
376.            }
377.        }
378.
379.        cout << "排序成功!(按总分降序排序)" << endl;
380.        cout << "(名次||姓名||性别||学号||班级||成绩.高数||成绩.程 C||成
    绩.离散||成绩.大物||总分||平均分)" << endl;
381.        cout << "-------------------------------------------------------
    ------------------------------" << endl;
382.        int number = 1;
383.        while (sortedList) {
384.            cout << number++ << " " << sortedList->getName() << " " <<
     sortedList->getGender() << " " << sortedList->getId()
385.                 << " " << sortedList->getClassName() << " ";
386.            const int *scores = sortedList->getScores();
387.            for (int i = 0; i < 4; ++i) {
388.                cout << scores[i] << ' ';
389.            }
390.            cout << sortedList->getTotalScore() << " " << sortedList->
    getAverageScore() << endl;
391.            // 移动到下一个节点并释放当前节点的内存
392.            Student *temp = sortedList;
```

```
393.           sortedList = sortedList->next;
394.           delete temp;
395.       }
396.       cout << "-------------------------------------------------------
   ------------------------------" << endl;
397.   }
398.
399.   void Manlist::managerSortStudentsByAverageScore(Stulist &stulistv)
    {
400.       Stulist *s = new Stulist(stulistv);
401.       Student *students = s->getHead()->next;
402.       // 插入排序按平均分降序排序
403.       Student *sortedList = nullptr;
404.       while (students) {
405.           // 从未排序链表中移除一个节点
406.           Student *current = students;
407.           students = students->next;
408.           // 将节点插入到已排序链表中的正确位置
409.           if (!sortedList || current->getAverageScore() > sortedList
   ->getAverageScore()) {
410.               // 将节点插入到已排序链表的开头
411.               current->next = sortedList;
412.               sortedList = current;
413.           } else {
414.               // 在已排序链表中找到插入位置
415.               Student *temp = sortedList;
416.               while (temp->next && temp->next->getAverageScore() > c
   urrent->getAverageScore()) {
417.                   temp = temp->next;
418.               }
419.               // 将节点插入到已排序链表的中间
420.               current->next = temp->next;
421.               temp->next = current;
422.           }
423.       }
424.
425.       cout << "排序成功!(按平均分降序排序)" << endl;
426.       cout << "(名次||姓名||性别||学号||班级||成绩.高数||成绩.程 C||成
   绩.离散||成绩.大物||总分||平均分)" << endl;
427.       cout << "-------------------------------------------------------
   ------------------------------" << endl;
428.       int number = 1;
429.       while (sortedList) {
```

```
430.            cout << number++ << " " << sortedList->getName() << " " <<
    sortedList->getGender() << " " << sortedList->getId()
431.                 << " " << sortedList->getClassName() << " ";
432.            const int *scores = sortedList->getScores();;
433.            for (int i = 0; i < 4; ++i) {
434.                cout << scores[i] << ' ';
435.            }
436.            cout << sortedList->getTotalScore() << " " << sortedList->
   getAverageScore() << endl;
437.            // 移动到下一个节点并释放当前节点的内存
438.            Student *temp = sortedList;
439.            sortedList = sortedList->next;
440.            delete temp;
441.        }
442.        cout << "-----------------------------------------------------
   ------------------------------" << endl;
443.    }
444.
445.    void Manlist::managerCountStudentsScore(Stulist &stulistv) {
446.        Stulist *s = new Stulist(stulistv);
447.        Student *students = s->getHead()->next;
448.
449.        map<string, vector<Student *>> classStudentsMap;
450.        while (students) {
451.            string className = students->getClassName();
452.            classStudentsMap[className].push_back(students);
453.            students = students->next;
454.        }
455.
456.        cout << "统计成功!" << endl;
457.        cout << "------------------------------" << endl;
458.
459.        vector<pair<string, double>> classAverages;
460.        for (auto &pair: classStudentsMap) {
461.            string className = pair.first;
462.            vector<Student *> &classStudents = pair.second;
463.
464.            double totalAverage = 0.0;
465.            int number = classStudents.size();
466.
467.            for (Student *student: classStudents) {
468.                totalAverage += student->getTotalScore();
469.            }
470.            totalAverage /= number;
```

```
471.
472.           classAverages.push_back(make_pair(className, totalAverage)
    );
473.       }
474.
475.       sort(classAverages.begin(), classAverages.end(), [](const pair
    <string, double> &a, const pair<string, double> &b) {
476.           return a.second > b.second;
477.       });
478.
479.       int rank = 1;
480.       for (const auto &classAverage: classAverages) {
481.           cout << "班级排
    名: " << rank++ << ". " << classAverage.first << "  平均
    分: " << classAverage.second << endl;
482.       }
483.
484.       cout << "-----------------------------" << endl;
485.
486.       for (auto &pair: classStudentsMap) {
487.           string className = pair.first;
488.           vector<Student *> &classStudents = pair.second;
489.
490.           Student *sortedList = nullptr;
491.           for (Student *current: classStudents) {
492.               if (!sortedList || current->getTotalScore() > sortedLi
    st->getTotalScore()) {
493.                   current->next = sortedList;
494.                   sortedList = current;
495.               } else {
496.                   Student *temp = sortedList;
497.                   while (temp->next && temp->next->getTotalScore() >
     current->getTotalScore()) {
498.                       temp = temp->next;
499.                   }
500.                   current->next = temp->next;
501.                   temp->next = current;
502.               }
503.           }
504.
505.           cout << className << "班学生成绩:" << endl;
506.           int number = 0;
507.           double totalAverage = 0.0;
508.           int numAbove60[4] = {0};
```

```
509.            double totalSubjectScores[4] = {0};
510.
511.            while (sortedList) {
512.                const int *scores = sortedList->getScores();
513.                for (int i = 0; i < 4; ++i) {
514.                    totalSubjectScores[i] += scores[i];
515.                    if (scores[i] > 60) {
516.                        numAbove60[i]++;
517.                    }
518.                }
519.                totalAverage += sortedList->getTotalScore();
520.
521.                Student *temp = sortedList;
522.                sortedList = sortedList->next;
523.                delete temp;
524.                number++;
525.            }
526.            cout << "班级总人数:" << number << endl << endl;
527.            cout << "各科成绩平均分:" << endl;
528.            const char *subjectNames[4] = {"高数", "程 C", "离散", "大物
    "};
529.            for (int i = 0; i < 4; ++i) {
530.                cout << subjectNames[i] << ": " << totalSubjectScores[
    i] / (number * 1.0) << endl;
531.            }
532.            cout << endl;
533.            cout << "各科及格的人数:" << endl;
534.            for (int i = 0; i < 4; ++i) {
535.                cout << subjectNames[i] << ": " << numAbove60[i] << en
    dl;
536.            }
537.            cout << endl;
538.            cout << "班级平均
    分:" << totalAverage / (number * 1.0) << endl;
539.            cout << "----------------------------" << endl;
540.        }
541.    }
542.
543.    void Manlist::managerUpdateStudentByID(Stulist &stulistv) {
544.        string idv;
545.        cout << "请输入您要修改学生信息的学号:" << endl;
546.        cin >> idv;
547.        Student *stu = stulistv.checkIfExist(idv);
548.        if (stu == nullptr) {
```

```
549.            cout << "很抱歉，并未找到该学生!" << endl;
550.            return;
551.        }
552.        cout << "----------------------------------" << endl;
553.        cout << "当前学生信息为:" << endl;
554.        cout << "姓名:" << stu->getName() << endl;
555.        cout << "性别:" << stu->getGender() << endl;
556.        cout << "学号:" << stu->getId() << endl;
557.        cout << "班级:" << stu->getClassName() << endl;
558.        cout << "成绩(高数,程C,离散,大物):";
559.        const int *scores = stu->getScores();
560.        for (int i = 0; i < 4; ++i) {
561.            cout << scores[i] << ' ';
562.        }
563.        cout << endl;
564.        cout << "总分:" << stu->getTotalScore() << endl;
565.        cout << "平均分:" << stu->getAverageScore() << endl;
566.        cout << "----------------------------------" << endl;
567.
568.        cout << "请输入修改后的信息(注意:如果信息不变请输入原来
    的!):" << endl;
569.        string namev, genderv, classNamev;
570.        int scoresv[4];
571.        cout << "姓名:" << endl;
572.        cin >> namev;
573.        cout << "性别:" << endl;
574.        cin >> genderv;
575.        if (!(genderv == "男" || genderv == "女")){
576.            cout << "性别输入无效!" << endl;
577.            return;
578.        }
579.        cout << "班级:" << endl;
580.        cin >> classNamev;
581.        cout << "成绩(高数,程C,离散,大物):" << endl;
582.        for (int i = 0; i < 4; ++i) cin >> scoresv[i];
583.
584.        stu->setName(namev);
585.        stu->setGender(genderv);
586.        stu->setClassName(classNamev);
587.        stu->readScores(scoresv);
588.
589.        stulistv.write();
590.        cout << "学生信息修改成功!" << endl;
591.        cout << "姓名:" << stu->getName() << endl;
```

```
592.        cout << "性别:" << stu->getGender() << endl;
593.        cout << "学号:" << stu->getId() << endl;
594.        cout << "班级:" << stu->getClassName() << endl;
595.        cout << "成绩(高数,程C,离散,大物):";
596.        for (int i = 0; i < 4; ++i) {
597.            cout << scores[i] << ' ';
598.        }
599.        cout << endl;
600.        cout << "----------------------------------" << endl;
601.    }
602.
603.    void Manlist::managerDeleteStudentByID(Stulist &stulistv) {
604.        string idv;
605.        cout << "请输入您要删除学生信息的学号:" << endl;
606.        cin >> idv;
607.
608.        Student *current = stulistv.getHead()->next;
609.        Student *pre = nullptr;
610.        while (current != nullptr) {
611.            if (current->getId() == idv) {
612.                if (pre == nullptr) {
613.                    stulistv.getHead()->next = current->next;
614.                } else {
615.                    pre->next = current->next;
616.                }
617.                delete current;
618.                stulistv.write();
619.                cout << "成功删除学生信息!" << endl;
620.                return;
621.            }
622.            pre = current;
623.            current = current->next;
624.        }
625.        cout << "很抱歉，并未找到该学生!" << endl;
626.    }
627.
628.
629.    void Manlist::delManager() {
630.        string usernamev;
631.        cout << "请输入您要删除教师账号的用户名:" << endl;
632.        cin >> usernamev;
633.        Manager *p = head->next, *pre = head;
634.        while (p) {
635.            if (p->getUsername() == usernamev) {
```

```
636.                 pre->next = p->next;
637.                 delete p;
638.                 size--;
639.                 write();
640.                 cout << "删除成功!" << endl;
641.                 return;
642.             }
643.             pre = p;
644.             p = p->next;
645.         }
646.         cout << "未找到该教师账号!" << endl;
647.     }
648.
649.     void Manlist::resetPassword() {
650.         string usernamev;
651.         cout << "请输入您要重置教师账号的用户名:" << endl;
652.         cin >> usernamev;
653.         Manager *man = checkIfExist(usernamev);
654.         if (man == nullptr) {
655.             cout << "未找到该教师账号!" << endl;
656.             return;
657.         }
658.         string passwordv1, passwordv2;
659.         cout << "请输入新密码:" << endl;
660.         cin >> passwordv1;
661.         cout << "请再次输入密码:" << endl;
662.         cin >> passwordv2;
663.         if (passwordv1 != passwordv2) {
664.             cout << "请输入相同的密码!" << endl;
665.             return;
666.         }
667.         man->setPassword(passwordv1);
668.         write();
669.         cout << "重置成功!" << endl;
670.     }
671.
672.     void Manlist::write() {
673.         ofstream out("../txt/Manager.txt");
674.         Manager *p = head->next;
675.         while (p) {
676.             out << p->getUsername() << ' ';
677.             out << p->getPassword() << ' ';
678.             out << p->getUserType() << ' ';
679.             out << p->getStatus() << ' ';
```

```
680.            out << p->getClassManage() << ' ' << endl;
681.            p = p->next;
682.        }
683.        out.close();
684.    }
685.
686.    bool Manlist::lockAccount(const string &usernamev) {
687.        Manager *target = findManagerByUsername(usernamev);
688.        if (target != nullptr) {
689.            target->setStatus(2);
690.            write();
691.            return true;
692.        }
693.        return false;
694.    }
695.
696.    void Manlist::setAccount() {
697.        string usernamev;
698.        cout << "请输入您要修改教师账号状态的用户名:" << endl;
699.        cin >> usernamev;
700.        Manager *man = checkIfExist(usernamev);
701.        if (man == nullptr) {
702.            cout << "未找到该教师账号!" << endl;
703.            return;
704.        }
705.        int statusv;
706.        cout << "请输入您要修改的状态(1:活跃 2:锁定):";
707.        cin >> statusv;
708.        man->setStatus(statusv);
709.        write();
710.        cout << "修改成功!" << endl;
711.    }
712.
713.    void Manlist::showAllManagers() {
714.        Manager *current = head->next;
715.        while (current != nullptr) {
716.            cout << "Username: " << current->getUsername() << endl;
717.            cout << "Password: " << current->getPassword() << endl;
718.            cout << "User Type: " << current->getUserType() << endl;
719.            cout << "Status: " << current->getStatus() << endl;
720.            cout << "ClassManage" << current->getClassManage() << endl
    ;
721.            cout << "------------------------" << endl;
722.
```

```
723.             current = current->next;
724.         }
725.     }
```