

程序设计课程设计报告

专 业:	计算机科学与技术
班 级:	计科 231
学 号:	32306100078
姓 名:	张瑾瑜

计算机科学与网络工程学院

二零二四年六月

一、 课程设计题目及内容

题目四：学生学籍管理系统

设计一个程序，对学生的学籍信息进行管理。通过该系统实现对学生基本信息的录入、保存、删除、修改、查询等操作。

设计要求及提示如下：

1、学生基本信息包括：学号、姓名、性别、年龄、班号、专业名称、系别等。

2、使用类和对象的概念实现程序设计。

3、以菜单形式显示各功能项。

4、各功能项如下：

(1) 学生基本信息的录入。

(2) 学生基本信息的删除。

(3) 学生基本信息的修改。

(4) 学生基本信息的查询。

(a) 按学号查询单个学生信息；

(b) 按姓名查询单个学生信息；

(c) 按性别查询学生信息；

(d) 按班号查询学生信息；

(5) 学生基本信息的统计。

(a) 按性别统计学生人数；

(b) 按班号统计学生人数；

(c) 按年龄统计学生人数；

(d) 按系别统计学生人数；

(6) 退出系统。

5、执行某个功能之后，程序将重新显示菜单供用户选择。

6、将学生学籍信息保存到文件中。

二、 功能模块的设计分析及算法描述

功能模块的设计分析

定义学生类

要求 1：学生基本信息包括：学号、姓名、性别、年龄、班号、专业名称、系别等。

设计分析：可以定义了一个名为 Student 的类，用于表示学生的基本信息。Student 类里可以定义构造函数去创建和初始化学生对象。由于姓名、性别、专业名称、系别中可能出现中文，并且学号和班号使用 string 类型可以避免使用一些类型转化和数据格式化的操作，因此我们定义其类型为 string 类型。

定义学生管理系统类

要求 2：通过该系统实现对学生基本信息的录入、保存、删除、修改、查询等操作

设计分析：可以定义一个学生管理系统类，用于封装对学生基本信息进行录入、保存、删除、修改、查询等操作，这样可以通过直接调用方法来简化代码。

要求 3: 可以利用系统进行学生基本信息的录入。学生的基本信息有学号、姓名、性别、年龄、班号、专业名称、系别等。

设计分析: 我们可以设计一个学生数组去储存学生信息。学号为 11 位数, 学生年龄为 15~50 岁 (考虑到有竞赛保送和唐尚郡这样的高龄高考钉子户入学), 班号为三位数 (例: 231 班, 前两位为入学年份), 因此我们需要对用户进行提示和限制。

因为需要一个数组大小对数组进行进行初始化, 因此在开始输入学生信息之前, 我们需要检查学生数量是否达到上限, 若达到上限则需要给出提示。

要求 4: 可以利用系统进行学生基本信息的删除。

设计分析: 学生的基本信息有学号、姓名、性别、年龄、班号、专业名称、系别等, 与个人关系强相关的两个信息分别是学号和姓名, 由于考虑到同一间学校中有出现同名的可能性, 而不会出现有两个相同学号的人, 因此只能设计成根据学号去删除学生。

考虑到刚点进删除选项的用户可能误触该选项或者不记得需要删除的学生的学号, 因此需要设计退出选择按钮。

考虑到删除学生信息是一个需要谨慎考虑的功能, 为了防止误删, 设计在输入学号后显示学生信息并再次询问用户是否删除的功能。

要求 5: 可以利用系统进行学生基本信息的修改。

设计分析: 首先需要检索到需要修改的学生, 检索到需要修改的学生的方式是**使用学号进行检索**, 原因同删除板块: 为了避免检索到多个同名的人并进行错误的修改。考虑可能会有误触的用户, 因此也需要提供返回主菜单的选项。

同删除板块, 修改学生信息也**需要用户确认**才不容易修改出错, 因此设计**在输入学号后显示学生信息并再次询问用户是否修改**的功能, 谨防修改出错。

考虑到**修改可能不止修改一个项**, 因此使用 `while` 循环设计子菜单, **提示用户输入需要修改的选项**, 这样可以简化用户修改流程。

要求 6: 可以利用系统进行学生基本信息的查询。可以按学号、姓名、性别、班号查询学生信息, 并按表格形式输出。

设计分析: 查询学生信息有**四个渠道**: 按学号、姓名、性别、班号查询, 因此设计子菜单, 让用户**选择需要的查询方式**, 再**调用相关的查询函数**进行查询。由于要求**按照表格形式输出**, 因此输出格式使用 `setw` 函数制表。

要求 7: 可以利用系统进行学生基本信息的统计。可以按性别、班号、年龄、系别统计学生信息。

设计分析: 统计学生信息有**四个渠道**: 按性别、班号、年龄、系

别查询，因此设计子菜单，让用户选择需要的统计方式，再调用相关的统计函数进行统计。

要求 8: 退出系统时将学生学籍信息保存到文件中。

设计分析: 由于要求退出系统功能，也要求将学籍信息保存在文件中，因此设计一开始启动程序时引导用户通过输入文件路径的方式打开指定文件，并对文件中的信息进行读取写入删除修改查询统计等操作，最后在退出的时候将信息写入文件内。考虑到有用户可能会输错文件路径，因此设计一个无限循环去提示用户输入正确的文件路径，当且仅当文件路径正确并能被打开时退出循环。

要求 9: 执行某个功能之后，程序将重新显示菜单供用户选择。

设计分析: 由于要求执行某个功能之后，程序将重新显示菜单供用户选择，因此用 while 循环设计主菜单，并在主菜单中根据用户选择调用各种子菜单，达到重新显示的目的。

算法描述

定义学生类

要求 1: 学生基本信息包括：学号、姓名、性别、年龄、班号、专业名称、系别等。

算法分析: 定义了一个包含学生基本信息的类 Student，定义

和初始化数据成员,并通过构造函数提供了两种初始化对象的方式:无参构造函数用于创建空的 `Student` 对象,而参数化构造函数允许通过传递所有必要的学生信息来初始化对象。这种设计使得可以轻松地创建和管理多个学生对象,并且可以方便地访问和修改每个学生的属性。

定义学生管理类

要求 2: 通过该系统实现对学生基本信息的录入、保存、删除、修改、查询等操作。

算法分析: 定义了 `StudentManagementSystem` 类实现了对学生基本数据进行操作的功能:使用了数组来存储学生对象,通过构造函数初始化了文件名和初始学生数量,并定义了许多对学生基本数据进行操作的公有成员方法,使得可以通过公有接口进行访问。

要求 3: 可以利用系统进行学生基本信息的录入。学生的基本信息有学号、姓名、性别、年龄、班号、专业名称、系别等。

算法分析: 在开始输入前为了防止数组溢出,会先检查当前学生数量 `studentCount` 是否已经达到预设的最大学生数量 `MAX_STUDENTS`,如果已经达到,则输出提示信息 "学生数量已满,无法添加更多学生。" 并直接返回,不再执行添加操作。如果未达到,则声明一个新的 `Student` 对象 `s`,用于存储即

将输入的学生信息。然后逐项输入学生信息。

特殊项判断算法:

(1) 学号限制为 11 位数的算法: 在 while 函数中使用 `length` 函数得出学号的位数, 若学号不为 11 位则提示用户重新输入并再次进行判断。

(2) 年龄为 15~50 岁之间的算法: 在 while 函数使用 “||” 作为限制条件, 当输入的数据不满足 15~50 岁时则提示用户重新输入, 并再次进行判断。

(3) 班号为三位数的算法: 同学号位数算法相同, while 循环中利用 `length` 函数进行位数的判别。

在所有信息录入成功后将新的学生对象添加到学生数组中并输出提示 “学生信息已添加”。

要求 4: 可以利用系统进行学生基本信息的删除。

算法分析: 在开始删除学生信息前, 考虑到可能学生数组还未录入学生信息, 为了防止误触删除按钮后产生的 bug, 先检查当前学生数量 `studentCount` 是否为 0, 如果没有学生信息 (即 `studentCount == 0`), 输出提示信息 “当前没有学生信息可供删除。” 并直接返回, 不再执行删除操作。

如果有学生信息, 则提示用户输入要删除的学生学号, 同时说明输入 0 可以返回主菜单。使用 `cin >> id`; 将用户输入存储在 `id` 变量中。

如果用户输入的学号是 "0", 则直接返回, 不进行删除操作。

如果不是 0, 则使用一个循环遍历 `students` 数组, 从索引 0 到 `studentCount - 1`。使用 `(students[i].id == id)` 检查每个学生的学号是否与用户输入的 `id` 相匹配。

如果找到匹配的学生, 将 `found` 标志设为 `true`。输出 "找到学号为 " << `id` <<" 的学生信息: " <<`endl`;; 并调用 `displayStudent(students[i])`; 函数显示该学生的信息。然后提示用户确认是否删除该学生信息, 输入 Y 或 N 作出选择。

如果用户确认要删除 (`confirm == 'Y' || confirm == 'y'`), 则使用一个循环将数组中从索引 `i` 开始的后续元素依次前移覆盖, 实现删除操作。然后减少 `studentCount` 计数器, 表示学生数量减少了一个。并输出 "学号为" << `id` << " 的学生信息已删除。" 表示删除成功。

如果用户选择取消删除操作, 则输出 "删除操作已取消。"。

如果在循环中没有找到与输入的学号匹配的学生, 输出 "未找到学号为 " << `id` << " 的学生。"。

要求 5: 可以利用系统进行学生基本信息的修改。

算法描述: 首先输出提示: "请输入要修改的学生学号 (输入 0 返回主菜单): " 然后读取用户输入的学号。如果输入为 0, 返回主菜单。

然后设置查找学生初始化变量 found 为 false。遍历学生列表：

如果学生的学号与输入学号匹配：设置 found 为 true。显示该学生的信息，然后跳出循环。

如果 found 仍为 false，则输出提示：“未找到该学号的学生。”，返回主菜单。

在 found 为 true 的情况下输出是否确认修改提示：“您确定要修改该学生的信息吗？（Y/N）：”然后读取用户输入。如果用户输入为 N 或 n，则输出提示：“修改操作已取消。”，然后返回主菜单。

如果用户输入为 Y 或 y，则输出提示菜单：“请选择要修改的信息（可多次选择，输入 0 返回主菜单）”，然后读取用户输入的选择，如果用户输入为 0，结束修改操作，返回主菜单。然后利用 switch 语句根据用户选择的修改项进行信息的修改。

在修改的过程中对输入的新信息进行合法性检查：年龄必须在 15 到 50 岁之间。班号必须是三位数。

最后更新学生记录中的对应字段，显示修改结果显示修改后的学生信息。然后输出提示：“信息修改成功。”然后返回主菜单。

要求 6: 可以利用系统进行学生基本信息的查询。可以按学号、姓名、性别、班号查询学生信息。

算法描述: 查询学生有四种方式，按学号、姓名、性别、班号：

按学号查询: 先提示用户输入学号，然后使用 for 循环遍历学生

信息列表, 找到与输入学号匹配的学生, 设置 found 标志为真, 接着显示该学生的信息。如果遍历结束后 found 标志为假, 则未找到匹配的学生, 提示用户未找到。

按姓名查询算法描述: 先提示用户输入姓名, 然后使用 for 循环遍历学生信息列表, 如果找到匹配姓名的学生, 显示学生信息, 设置 found 标志为真。如果遍历结束后 found 标志为假, 提示未找到。

按性别查询算法描述: 先提示用户输入性别, 然后使用 for 循环遍历学生列表。如果找到匹配性别的学生, 显示学生信息, 设置 found 标志为真。如果遍历结束后 found 标志为假, 提示未找到。

按班号查询算法描述: 先提示用户输入班号。然后使用 for 循环遍历学生列表。如果找到匹配班号的学生, 显示学生信息, 设置 found 标志为真。如果遍历结束后 found 标志为假, 提示未找到。

要求 7: 可以利用系统进行学生基本信息的统计。可以按性别、班号、年龄、系别统计学生信息。

算法描述: 先定义结构体 Department, 它包含两个成员变量, 其中 name 表示系别名称, count 表示该系别的学生人数统计。然后声明一个数组 departments, 用来存储各个系别的学生人数统计信息。

初始化一个计数器 `departmentCount`, 用来记录不同系别的数量。

使用 `for` 循环遍历所有学生, 索引从 0 到 `studentCount - 1`。对于每个学生初始化一个布尔变量 `found` 为 `false`, 用于标记是否找到与当前学生系别相同的记录。然后再次使用 `for` 循环遍历 `departments` 数组, 索引从 0 到 `departmentCount - 1`。

如果找到某个 `department[j].name` 等于当前学生的 `students[i].department`, 表示该系别已经在 `departments` 数组中统计过, 接着将该系别的 `count` 加一 (`departments[j].count++`)。将 `found` 设置为 `true`, 表示找到了已存在的记录, 接着跳出内部循环, 继续处理下一个学生。

如果内部循环结束时 `found` 仍然为 `false`, 表示当前学生的系别是一个新的系别, 尚未在 `departments` 数组中统计过。

然后将新的系别名称 `students[i].department` 存储在 `departments[departmentCount].name` 中。并将该系别的人数统计 `departments[departmentCount].count` 初始化为 1。

接着增加 `departmentCount` 计数器, 表示新增了一个新的系别统计。

最后使用 `for` 循环遍历 `departments` 数组, 索引从 0 到 `departmentCount - 1`, 输出每一个系别名称和对应的学生人

数统计.

要求 8: 退出系统时将学生学籍信息保存到文件中。

算法描述:

从文件中加载学生信息算法描述: 首先使用 `while (true)` 创建一个无限循环, 用于**反复尝试打开文件**。

每次循环中, 尝试使用 `file.open(filename)` 打开指定文件。如果文件无法打开 (`if (!file.is_open())`), **提示用户重新输入文件路径**。如果文件成功打开, **使用 `break` 跳出循环**, 继续后续操作。

在成功打开文件后, 初始化相关变量, 并**重置学生计数器 `studentCount`**。**设置预览的学生信息数量 `previewCount`**, 用于**显示前几条学生信息供用户确认**。然后使用 `while (file >> id >> name >> gender >> age >> classId >> major >> department)` 循环读取文件内容, 每次读取一行学生信息, 并**存储到临时数组 `tempStudents` 中**。

检查学生数量是否达到最大值 `MAX_STUDENTS`, 如已达到, 输出警告信息并停止读取。显示前 `previewCount` 条学生信息供用户确认。

读取完成后, 关闭文件 `file.close()`。接着**显示前 5 条学生信息**, 并**询问用户是否确认导入这些信息**。

如果用户确认信息正确, 则将临时存储的学生信息转移到正式

数组 `students` 中，并输出加载完成的信息。

如果用户确认信息不正确，则提示重新输入文件路径，并递归调用 `loadFromFile()` 重新加载文件，同时清空临时数组和学生计数器 `studentCount`。

保存学生信息到文件算法描述：在成功打开文件后，使用 `for` 循环遍历学生信息列表，将每个学生的信息按顺序写入文件，每个字段之间用空格分隔，每个学生信息占一行。写入完成后，使用 `file.close()` 函数关闭文件，并输出保存成功的提示。

要求 9：执行某个功能之后，程序将重新显示菜单供用户选择。

算法描述：使用 `while (true)` 构建一个无限循环，使得菜单能够持续显示直到用户选择退出，每次循环开始时，会打印一个菜单，列出所有选项，用户被提示输入选择，程序通过 `cin >> choice;` 获取用户的选择，然后再使用 `switch` 语句根据用户选择调用相关函数进行执行

三、 程序中使用的数据及主要符号说明

数据结构和变量

Student 类

用于表示单个学生的基本信息：

id: 学号 (`string` 类型, 11 位数字)

name: 姓名 (`string` 类型)

gender: 性别 (string 类型, 男或女)
age: 年龄 (int 类型, 15-50 岁之间)
classId: 班号 (string 类型, 3 位数字)
major: 专业名称 (string 类型)
department: 系别 (string 类型)

StudentManagementSystem 类

students: 存储学生信息的数组 (最大容量为 100)
studentCount: 当前学生数量 (int 类型)
filename: 保存学生信息的文件名 (string 类型)
addStudent(): 添加学生信息
deleteStudent(): 删除学生信息
modifyStudent(): 修改学生信息
searchStudent(): 查询学生信息
statistics(): 统计学生信息
loadFromFile(): 从文件加载学生信息
saveToFile(): 保存学生信息到文件
displayAllStudents(): 显示所有学生信息
menu(): 显示主菜单并处理用户输入

主要符号说明

类定义

```
class Student { ... };
```

定义了一个学生类, 包含学生的基本信息字段和构造函数。

```
class StudentManagementSystem { ... };
```

定义了一个学生管理系统类, 包含学生管理的各种方法和数据成员。

对象

`students[MAX_STUDENTS]`: 数组, 用于存储所有学生的信息。

`studentCount`: 当前已存储的学生数量。

`filename`: 保存和加载学生信息的文件名。

`modifiedStudents`: 列表, 用于存储本次操作中新添加、修改或删除的学生信息。

方法

`addStudent()`: 添加学生信息的方法, 要求输入学生的所有基本信息, 并进行合法性检查。

`deleteStudent()`: 删除学生信息的方法, 要求输入学号, 并确认删除操作。

`modifyStudent()`: 修改学生信息的方法, 要求输入学号,

找到该学生后允许用户选择需要修改的字段并进行修改。

`searchStudent()`：查询学生信息的方法，提供按学号、姓名、性别和班号查询的功能。

`statistics()`：统计学生信息的方法，按性别、班号、年龄和系别统计学生人数。

`loadFromFile()`：从文件加载学生信息的方法，读取文件内容并填充到学生数组中。

`saveToFile()`：保存学生信息到文件的方法，将当前学生数组中的信息写入文件。

`displayAllStudents()`：显示所有学生信息的方法，以表格形式输出所有学生的信息。

`menu()`：显示主菜单的方法，提供各种功能的入口并处理用户输入。

输入/输出

`cin`：用于从用户输入获取数据。

`cout`：用于向用户输出信息。

`ifstream`：用于从文件读取数据。

`ofstream`：用于向文件写入数据。

四、 部分关键程序的源代码

添加学生信息源代码：

```
// 添加学生信息的方法  
void addStudent() {
```

```
// 检查学生数量是否已达上限
if (studentCount >= MAX_STUDENTS) {
    cout << "学生数量已满，无法添加更多学生。" << endl;
    return;
}

Student s; // 创建一个新的学生对象

// 输入学号
cout << "请输入学号（11位数字）：";
cin >> s.id;
while (s.id.length() != 11) { // 验证学号长度为11位数字
    cout << "学号必须为11位数字，请重新输入：";
    cin >> s.id;
}

// 输入姓名
cout << "请输入姓名：";
cin >> s.name;

// 输入性别
cout << "请输入性别（男/女）：";
cin >> s.gender;

// 输入年龄
cout << "请输入年龄（15-50）：";
cin >> s.age;
while (s.age < 15 || s.age > 50) { // 验证年龄在15到50岁之间
    cout << "年龄必须在15到50岁之间，请重新输入：";
    cin >> s.age;
}

//输入班号
cout << "请输入班号（三位数）：";
cin >> s.classId;
while (s.classId.length() != 3) { // 验证班号长度为3位数字
    cout << "班号必须为三位数字，请重新输入：";
    cin >> s.classId;
}

// 输入专业名称
cout << "请输入专业名称：";
cin >> s.major;
```

```

// 输入系别
cout << "请输入系别：";
cin >> s.department;

// 将新学生对象添加到学生数组中
students[studentCount++] = s;

cout << "学生信息已添加。" << endl;
}

```

修改学生信息源代码：

```

// 修改学生信息的方法
void modifyStudent() {
    string id;
    cout << "请输入要修改的学生学号（输入0返回主菜单）：";
    cin >> id;

    if (id == "0") {
        return; // 返回主菜单
    }

    bool found = false;

    // 遍历学生数组，查找匹配的学生
    for (int i = 0; i < studentCount; ++i) {
        if (students[i].id == id) {
            found = true;
            // 显示该学生的信息
            cout << "找到学号为 " << id << " 的学生信息：" << endl;
            displayStudent(students[i]);

            char confirm;
            cout << "您确定要修改该学生的信息吗？（Y/N）：";
            cin >> confirm;

            if (confirm == 'Y' || confirm == 'y') {
                int choice;
                while (true) {
                    cout << "\n请选择要修改的信息（可多次选择，输入0结束）：" << endl;
                    cout << "1. 姓名" << endl;
                    cout << "2. 性别" << endl;
                    cout << "3. 年龄" << endl;
                    cout << "4. 班号" << endl;
                    cout << "5. 专业名称" << endl;

```

```
cout << "6. 系别" << endl;
cout << "请输入选择: ";
cin >> choice;

if (choice == 0) break;

switch (choice) {
case 1:
    // 修改姓名
    cout << "请输入新的姓名: ";
    cin >> students[i].name;
    break;

case 2:
    // 修改性别
    cout << "请输入新的性别: ";
    cin >> students[i].gender;
    break;

case 3:
    // 修改年龄, 确保在15到50岁之间
    while (true) {
        cout << "请输入新的年龄 (15~50岁): ";
        cin >> students[i].age;
        if (students[i].age >= 15 && students[i].age <= 50) break;
        cout << "年龄必须在15到50岁之间, 请重新输入." << endl;
    }
    break;

case 4:
    // 修改班号
    while (true) {
        cout << "请输入新的班号 (3位数字): ";
        cin >> students[i].classId;
        while (students[i].classId.length() != 3) {
            cout << "班号必须为3位数字, 请重新输入: ";
            cin >> students[i].classId;
        }
        break;
    }
    break;

case 5:
    // 修改专业名称
    cout << "请输入新的专业名称: ";
```

```

        cin >> students[i].major;
        break;
    case 6:
        // 修改系别
        cout << "请输入新的系别: ";
        cin >> students[i].department;
        break;
    default:
        cout << "无效选择, 请重试。" << endl;
        continue; // 重新进入选择循环
    }

    cout << "信息修改成功。" << endl;
}

return;
}

else {
    cout << "修改操作已取消。" << endl;
}

break; // 已经找到并处理了该学生信息, 退出循环
}

}

if (!found) {
    cout << "未找到该学号的学生。" << endl;
}

}

```

删除学生信息源代码:

```

// 删除学生信息
void deleteStudent() {
    if (studentCount == 0) {
        cout << "当前没有学生信息可供删除。" << endl;
        return;
    }

    string id;
    cout << "请输入要删除的学生学号 (输入0返回主菜单): ";
    cin >> id;

    if (id == "0") {
        return; // 返回主菜单
    }

    bool found = false;

```

```

for (int i = 0; i < studentCount; ++i) {
    if (students[i].id == id) {
        found = true;
        // 显示该学生的信息
        cout << "找到学号为 " << id << " 的学生信息：" << endl;
        displayStudent(students[i]);

        char confirm;
        cout << "您确定要删除该学生的信息吗？(Y/N)："；
        cin >> confirm;

        if (confirm == 'Y' || confirm == 'y') {
            // 删除该学生，将后面的学生信息前移
            for (int j = i; j < studentCount - 1; ++j) {
                students[j] = students[j + 1];
            }
            studentCount--;
            cout << "学号为 " << id << " 的学生信息已删除。" << endl;
        }
        else {
            cout << "删除操作已取消。" << endl;
        }
        break;
    }
}

if (!found) {
    cout << "未找到学号为 " << id << " 的学生。" << endl;
}
}

```

查询学生信息源代码（以使用班号查询为例）：

```

// 按班号查询学生信息
void searchByClassId() {
    string classId;
    cout << "请输入要查询的班号："；
    cin >> classId;

    // 查找学生并输出信息
    bool found = false;
    for (int i = 0; i < studentCount; ++i) {
        if (students[i].classId == classId) {
            if (!found) {
                cout << "\n查询结果：" << endl;
                cout << setw(12) << left << "学号" << setw(12) << left << "姓名" <<

```

```

setw(6) << left << "性别"
                << setw(6) << left << "年龄" << setw(6) << left << "班号" <<
setw(24) << left << "专业"
                << setw(6) << left << "系别" << endl;
    }
    displayStudent(students[i]);
    found = true;
}
}

if (!found) {
    cout << "未找到班号为 " << classId << " 的学生。" << endl;
}
}

```

统计学生信息源代码（以用年龄统计为例）：

```

void statisticsByAge() {
    int ageCounts[51] = { 0 }; // 年龄范围 0~50

    // 统计各个年龄段的学生人数
    for (int i = 0; i < studentCount; ++i) {
        if (students[i].age >= 0 && students[i].age <= 50) {
            ageCounts[students[i].age]++; // 对应年龄段的学生人数加一
        }
    }

    // 输出年龄段学生人数统计结果（只统计合法年龄范围 15~50）
    for (int i = 15; i <= 50; ++i) {
        if (ageCounts[i] > 0) {
            cout << i << " 岁的学生人数: " << ageCounts[i] << endl;
        }
    }
}
}

```

输出学生信息源代码：

```

void statisticsByAge() {
    int ageCounts[51] = { 0 }; // 年龄范围 0~50

    // 统计各个年龄段的学生人数
    for (int i = 0; i < studentCount; ++i) {
        if (students[i].age >= 0 && students[i].age <= 50) {
            ageCounts[students[i].age]++; // 对应年龄段的学生人数加一
        }
    }
}

```

```

// 输出年龄段学生人数统计结果（只统计合法年龄范围 15~50）
for (int i = 15; i <= 50; ++i) {
    if (ageCounts[i] > 0) {
        cout << i << " 岁的学生人数: " << ageCounts[i] << endl;
    }
}
}

```

从用户输入的指定路径中加载文件源代码：

```

void loadFromFile() {
    ifstream file;
    while (true) {
        file.open(filename); // 尝试打开文件
        if (!file.is_open()) { // 如果文件无法打开
            cout << "无法打开文件，请重新输入文件路径: ";
            cin >> filename; // 让用户重新输入文件路径
        }
        else {
            break; // 文件成功打开，跳出循环
        }
    }

    string id, name, gender, classId, major, department;
    int age;
    studentCount = 0; // 重置学生计数
    int previewCount = 5; // 预览前几条学生信息
    int count = 0; // 记录实际预览的学生数量

    // 临时存储读取到的学生信息，用于用户确认
    Student tempStudents[MAX_STUDENTS];

    // 循环读取文件中的学生信息
    while (file >> id >> name >> gender >> age >> classId >> major >> department) {
        if (studentCount >= MAX_STUDENTS) { // 检查学生数量是否已达到最大值
            cout << "学生数量已达到最大值，无法加载更多学生信息。" << endl;
            break;
        }
        tempStudents[studentCount++] = Student(id, name, gender, age, classId, major,
        department);

        if (count < previewCount) {
            cout << "学生信息 " << count + 1 << ": ";
            cout << id << " " << name << " " << gender << " "

```



```

        << age << " " << classId << " " << major << " " << department << endl;
        ++count;
    }
}

file.close(); // 关闭文件

// 询问用户导入的信息是否正确
char confirm;
cout << "以上为文件中的前 " << count << " 条学生信息，您确定导入这些信息吗？
(Y/N) : ";
cin >> confirm;

if (confirm == 'Y' || confirm == 'y') {
    // 如果用户确认信息正确，则将临时存储的学生信息转移到正式数组中
    for (int i = 0; i < studentCount; ++i) {
        students[i] = tempStudents[i];
    }
    cout << "学生信息加载完毕，共加载 " << studentCount << " 条信息。" << endl;
}
else {
    // 如果用户确认信息不正确，则清空临时数组和学生计数器，并重新输入文件路径
    cout << "导入操作已取消，请重新输入文件路径。" << endl;
    studentCount = 0; // 重置学生计数
    filename = ""; // 清空文件名
    loadFromFile(); // 递归调用重新加载文件
}
}
}

```

退出并保存文件源代码：

```

void saveToFile() {
    ofstream file;
    while (true) {
        file.open(filename); // 尝试打开文件
        if (!file.is_open()) { // 如果文件无法打开
            cout << "无法打开文件，请重新输入文件路径：";
            cin >> filename; // 让用户重新输入文件路径
        }
        else {
            break; // 文件成功打开，跳出循环
        }
    }
}

// 遍历学生数组，将每个学生的信息写入文件

```

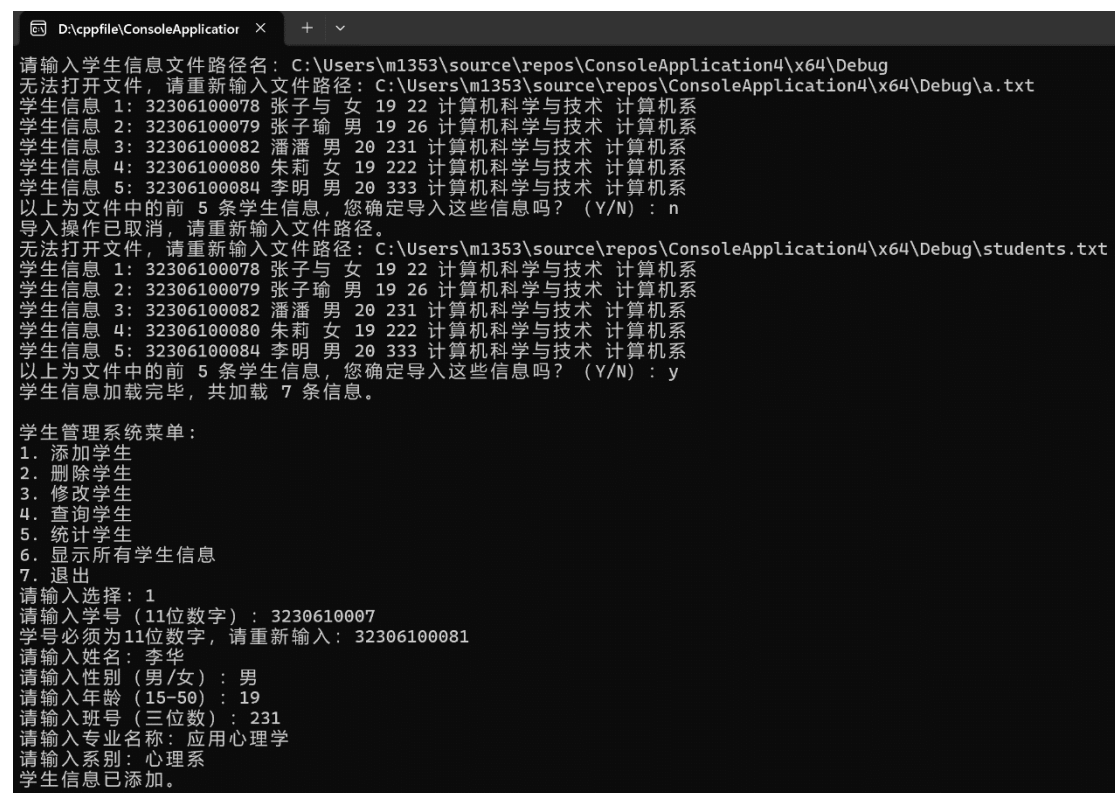
```

        for (int i = 0; i < studentCount; ++i) {
            file << students[i].id << " " << students[i].name << " " << students[i].gender
            << " "
                << students[i].age << " " << students[i].classId << " " <<
students[i].major << " "
                << students[i].department << endl;
        }

    file.close(); // 关闭文件
    cout << "学生信息已保存到文件。" << endl;
}

```

程序运行时的效果图



```

D:\cppfile\ConsoleApplication  ×  +  ▾
请输入学生信息文件路径名: C:\Users\m1353\source\repos\ConsoleApplication4\x64\Debug
无法打开文件, 请重新输入文件路径: C:\Users\m1353\source\repos\ConsoleApplication4\x64\Debug\a.txt
学生信息 1: 32306100078 张子与 女 19 22 计算机科学与技术 计算机系
学生信息 2: 32306100079 张子瑜 男 19 26 计算机科学与技术 计算机系
学生信息 3: 32306100082 潘潘 男 20 231 计算机科学与技术 计算机系
学生信息 4: 32306100080 朱莉 女 19 222 计算机科学与技术 计算机系
学生信息 5: 32306100084 李明 男 20 333 计算机科学与技术 计算机系
以上为文件中的前 5 条学生信息, 您确定导入这些信息吗? (Y/N): n
导入操作已取消, 请重新输入文件路径。
无法打开文件, 请重新输入文件路径: C:\Users\m1353\source\repos\ConsoleApplication4\x64\Debug\students.txt
学生信息 1: 32306100078 张子与 女 19 22 计算机科学与技术 计算机系
学生信息 2: 32306100079 张子瑜 男 19 26 计算机科学与技术 计算机系
学生信息 3: 32306100082 潘潘 男 20 231 计算机科学与技术 计算机系
学生信息 4: 32306100080 朱莉 女 19 222 计算机科学与技术 计算机系
学生信息 5: 32306100084 李明 男 20 333 计算机科学与技术 计算机系
以上为文件中的前 5 条学生信息, 您确定导入这些信息吗? (Y/N): y
学生信息加载完毕, 共加载 7 条信息。

学生管理系统菜单:
1. 添加学生
2. 删除学生
3. 修改学生
4. 查询学生
5. 统计学生
6. 显示所有学生信息
7. 退出
请输入选择: 1
请输入学号 (11位数字): 3230610007
学号必须为11位数字, 请重新输入: 32306100081
请输入姓名: 李华
请输入性别 (男/女): 男
请输入年龄 (15-50): 19
请输入班号 (三位数): 231
请输入专业名称: 应用心理学
请输入系别: 心理系
学生信息已添加。

```

请选择要修改的信息（可多次选择，输入0结束）：

1. 姓名
2. 性别
3. 年龄
4. 班号
5. 专业名称
6. 系别

请输入选择：1

请输入新的姓名：张子瑜

信息修改成功。

请选择要修改的信息（可多次选择，输入0结束）：

1. 姓名
2. 性别
3. 年龄
4. 班号
5. 专业名称
6. 系别

请输入选择：4

请输入新的班号（3位数字）：231

信息修改成功。

学生管理系统菜单：

1. 添加学生
2. 删除学生
3. 修改学生
4. 查询学生
5. 统计学生
6. 显示所有学生信息
7. 退出

请输入选择：4

查询学生信息：

1. 按学号查询
2. 按姓名查询
3. 按性别查询
4. 按班号查询
0. 返回主菜单

请输入选择：4

请输入要查询的班号：231

查询结果：

学号	姓名	性别	年龄	班号	专业	系别
32306100082	潘潘	男	20	231	计算机科学与技术	计算机系
32306100080	张子瑜	女	19	231	计算机科学与技术	计算机系
32306100078	张瑾瑜	女	25	231	计算机	计算机系

学生管理系统菜单：

1. 添加学生
2. 删除学生
3. 修改学生
4. 查询学生
5. 统计学生
6. 显示所有学生信息
7. 退出

请输入选择：5

统计学生信息子菜单（输入0返回主菜单）：

1. 按性别统计
2. 按班号统计
3. 按年龄统计
4. 按系别统计

请输入选择：1

男生人数：3

女生人数：4

学生管理系统菜单：

1. 添加学生
2. 删除学生
3. 修改学生
4. 查询学生
5. 统计学生
6. 显示所有学生信息
7. 退出

请输入选择：6

所有学生信息：

32306100078	张子与	女	19	22	计算机科学与技术	计算机系
32306100079	张子瑜	男	19	26	计算机科学与技术	计算机系
32306100082	潘潘	男	20	231	计算机科学与技术	计算机系
32306100080	张子瑜	女	19	231	计算机科学与技术	计算机系
32306100084	李明	男	20	333	计算机科学与技术	计算机系
32306100086	李子涵	女	21	352	应用心理学	心理系
32306100078	张瑾瑜	女	25	231	计算机	计算机系

查询学生信息：

1. 按学号查询
2. 按姓名查询
3. 按性别查询
4. 按班号查询
0. 返回主菜单

请输入选择：2

请输入要查询的学生姓名：张子瑜

查询结果：

学号	姓名	性别	年龄	班号	专业	系别
32306100079	张子瑜	男	19	26	计算机科学与技术	计算机系
32306100080	张子瑜	女	19	231	计算机科学与技术	计算机系

五、实验结果分析，实验收获和体会

5.1 实验结果分析

5.1.1 功能测试结果

学生信息添加

成功输入正确格式的信息（学号为 11 位数字，班号为 3 位数字，年龄在 15-50 岁之间）后，系统能够正确地将学生信息添加到系统中。输入错误格式的信息（如学号不是 11 位，班号不是 3 位，年龄超出范围）时，系统能够提示用户重新输入。

5.1.2 学生信息删除

输入正确的学号后，系统能够显示该学号对应的学生信息，并询问用户是否确认删除。用户确认后，系统能够成功删除该学生信息。

输入不存在的学号时，系统能够提示未找到该学号的学生信息。

5.1.3 学生信息修改

输入正确的学号后，系统能够显示该学号对应的学生信息，并允许用户选择需要修改的字段进行修改。修改完成后，系统能够正确保存修改后的信息。

修改过程中输入错误格式的信息时，系统能够提示用户重新输入。

5.1.4 学生信息查询

系统提供按学号、姓名、性别和班号查询的功能。查询结果能够以表格形式整齐显示，方便用户查看。

各种查询方式均能够返回正确的结果，用户体验较好。

5.1.5 学生信息统计

系统能够按性别、班号、年龄和系别统计学生人数，并以表格形式输出结果。统计结果准确，能够有效反映学生分布情况。

5.1.6 文件加载和保存

系统能够从用户输入的指定路径的文件中加载学生信息，并在修改后将信息保存到文件中。加载过程中显示前几条信息供用户确认，增强了用户的体验。

文件打开失败时，系统能够提示用户重新输入文件路径，保证数据读取的正确性。

5.2 实验收获与体会

在这次实验中，我通过一个比较完整的学生学籍管理系统应用程序的设计，将学过的知识连贯起来，在实践中掌握了开发一个实际应用程序的步骤，同时我也学会了从用户的角度去思考一个应用该怎么开发，学会了使用开发工具实现界面友好的应用程序。

在这个实验过程中，我初步理解面向对象的特性，掌握了一定的面向对象程序设计的开发方法，提升了分析问题和解决问题的能力。

六、 自评成绩

97 分，我的代码已经尽量做到简洁，而且我尝试把注释写得比较详细，并且根据实验要求认真的实现了所有的要求，在此基础上，我根据张艳玲老师的建议去尽量的想象了用户的使用场景，并且以“设计软件”的思维去对我的系统进行了许多提升用户体验的细节性适配（比如说为防止导入出错会显示前几行学籍名单、防止修改出错会显示检索出的学生名单并再次询问是否修改、通过一个循环可以自由的选择某一学生的不同修改项等等），但是我的代码算法复杂度不是很高，所以我认为当我掌握了关于算法的更多知识之后也许我能拿出更加优秀的作品。