

《面向对象程序设计》课程设计

题目：学生管理系统

年级：24级计算机

开发组序号：第X组

开发组情况：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 姓名 | 学号 | 班级 | 分工 | 自评分 |
| 陈康瑜 | 202311070419 | 2班 | 数据结构设计以及文件流存储 | 30% |
| 张棕源 | 202411040240 | 2班 | 主界面的图形化及功能实现 | 20% |
| 王昆 | 202311070414 | 2班 | 子界面的图形化及功能实现 | 25% |
| 刘峻辰 | 202311070415 | 2班 | 子界面的图形化及功能实现及功能实现 | 25% |

指导教师：王倩

开发日期：2025.6.3

## 摘要

**[摘 要]**

## 本文主要分为摘要、前言、目录、正文、参考文献五个部分。本报告主要记录了本小组在面向对象程序课程设计这门课中学生管理系统的具体设计方案、项目效果及心得体会。学生管理系统是生活中常见的一种信息管理系统设计，具有广泛的应用。本小组尝实现通过从c++实现数据结构设计，Qt辅助完成图形化界面设计，具有基本的增删改查功能之外，还具有成绩计算功能、成绩排序功能、文件流存储功能等。

**[关键词]**

C++ Qt 图形化界面 文件流操作 人机交互

## 前言

## 学生管理系统是高校信息化管理的重要工具，用于高效管理学生信息和课程成绩。本项目基于 C++ 语言和 Qt框架开发。项目的开发不仅加深了我们对 C++ 编程、图形界面设计和文件操作的理解，还培养了规范编码、文档编写和系统测试的实践经验。

## 本项目的目标是实现一个功能完善、界面友好的学生管理系统，支持课程录入、学生信息查询、宿舍查询、成绩查询、学生删除和成绩排序等功能。项目开发历时一周，完成了需求分析、系统设计、编码实现、测试优化和文档编写。开发过程中，我们克服了文件解析、输入验证和界面适配等技术难点，实现了背景适配和友好错误提示等创新功能。

## 通过本项目，我们不仅基本了解和使用了 C++ 和 Qt 的开发技能，还深刻体会到团队协作和规范开发的重要性。希望本报告能全面展示我们的努力与成果，为后续学习和开发提供参考。

## 目录

[摘要 2](#_Toc30516)

[前言 2](#_Toc8035)

[正文 5](#_Toc24070)

[1 概述 5](#_Toc21503)

[1.1. 目的与意义 5](#_Toc27674)

[1.2. 主要完成的任务 5](#_Toc21224)

[1.3. 解决的主要问题 6](#_Toc31120)

[1.4人员分工 6](#_Toc31505)

[1.5开发计划 6](#_Toc16910)

[1.6开发环境 7](#_Toc14141)

[2 总体设计 7](#_Toc1145)

[2.1 软件的总体结构 7](#_Toc24578)

[2.2 模块功能及关系 8](#_Toc30222)

[数据管理模块（StudentManager） 8](#_Toc26341)

[2.3 总体流程设计 10](#_Toc13467)

[3 详细设计 11](#_Toc32569)

[3.1 数据管理模块 11](#_Toc1740)

[3.1.1 模块流程和实现算法 11](#_Toc29116)

[3.1.2 类、对象、函数描述 12](#_Toc17444)

[3.2 主界面模块 14](#_Toc17292)

[3.2.1 模块流程和实现算法 14](#_Toc12286)

[3.2.2 类、对象、函数描述 15](#_Toc8585)

[3.3 子界面模块 - 录入课程 15](#_Toc2293)

[3.3.1 模块流程和实现算法 15](#_Toc15479)

[3.3.2 类、对象、函数描述 16](#_Toc19156)

[3.4 子界面模块 - 查询学生 16](#_Toc12926)

[3.4.1 模块流程和实现算法 16](#_Toc23275)

[3.4.2 类、对象、函数描述 17](#_Toc15447)

[3.5 子界面模块 - 查询宿舍 17](#_Toc23848)

[3.5.2 类、对象、函数描述 18](#_Toc27622)

[3.6 子界面模块 - 查询成绩 18](#_Toc27341)

[3.6.1 模块流程和实现算法 18](#_Toc12453)

[3.6.2 类、对象、函数描述 18](#_Toc2976)

[3.7 子界面模块 - 删除学生 19](#_Toc22680)

[3.7.1 模块流程和实现算法 19](#_Toc28067)

[3.7.2 类、对象、函数描述 19](#_Toc14746)

[3.8 子界面模块 - 排序成绩 19](#_Toc30625)

[3.8.1 模块流程和实现算法 19](#_Toc20026)

[3.8.2 类、对象、函数描述 20](#_Toc29970)

[4 编码设计 21](#_Toc10899)

[4.1 开发环境的设置和建立 21](#_Toc19999)

[4.2 程序设计时要注意的事项 22](#_Toc4284)

[4.3 关键构件/插件的特点和使用 22](#_Toc5331)

[4.4 主要程序的代码设计及注释 23](#_Toc27066)

[4.4.1 数据管理模块（StudentManager::addCourse） 23](#_Toc14516)

[4.4.2 主界面模块（MainWindow::setupUI） 24](#_Toc20262)

[4.5 解决的技术难点及经常犯的错误 27](#_Toc31843)

[4.5.1 技术难点 27](#_Toc29411)

[4.5.2 常见错误 27](#_Toc25839)

[4.6 特色算法示例 28](#_Toc23956)

[4.6.2 算法 2：学生查询（findStudentsByIdOrName） 29](#_Toc22779)

[5 测试时出现过的问题及其解决方法 30](#_Toc436)

[5.1 文件读写问题 30](#_Toc5614)

[5.2 背景图片加载失败 30](#_Toc20606)

[5.3 输入验证不足 31](#_Toc5058)

[5.4 查询结果为空 31](#_Toc9993)

[5.5 删除学生后文件未更新 31](#_Toc29743)

[6. 总结 32](#_Toc23869)

[6.1 课程设计的完成情况 32](#_Toc16450)

[6.2 创新功能 33](#_Toc19862)

[6.5 未来改进方向（拓展） 34](#_Toc4003)

[7. 附录 34](#_Toc9163)

[7.1参考文献 34](#_Toc20248)

[7.2源代码 34](#_Toc29135)

## 

## 正文

## 1 概述

**1.1. 目的与意义**

本系统旨在开发一个简易高效的学生管理系统，实现学生信息与成绩的数字化管理。通过自动化处理成绩计算、信息查询等功能，减少人工操作误差，提高学校教学管理效率。系统支持数据的录入、查询、删除和排序等核心功能，提供了便捷的学生信息管理工具，促进教学管理工作的规范化和信息化。

**1.2. 主要完成的任务**

数据处理模块：实现学生基本信息和课程成绩信息的加载与保存，支持数据的持久化存储。

成绩计算引擎：根据课程是否包含实验成绩，自动计算综合成绩。

学生基本情况查询：支持按学号 / 姓名精确查询，按宿舍号批量查询。

课程成绩查询：输入学号可查看所有课程的综合成绩和实得学分，并统计总修课程数与总学分。

课程信息录入：支持添加新的课程成绩记录，自动计算综合成绩与实得学分。

学生信息删除：删除学生时自动同步删除其所有课程记录，保证数据一致性。

排序功能：支持按综合成绩或实得学分进行升序 / 降序排序，便于成绩分析。

用户界面：设计直观的图形界面，通过按钮导航实现各功能模块的交互操作。

**1.3. 解决的主要问题**

**精确查询问题：**

实现高效的查询机制，支持通过学号或姓名快速查找到学生基本信息，解决多条件检索需求。

优化宿舍号查询逻辑，通过建立宿舍 - 学生映射关系，实现批量学生信息的快速提取与展示。

**成绩统计与展示：**

设计合理的数据结构存储课程信息和学生信息，自动计算综合成绩和实得学分，确保计算逻辑准确无误。

实现格式化输出功能，按指定格式展示学生课程信息，并自动统计总课程数和总学分。

**数据一致性维护**：

建立学生信息与课程成绩的关联机制，在删除学生记录时，同步删除关联课程数据，避免数据残留。

实现文件操作的异常处理，确保数据删除过程中的完整性。

**数据排序与分析：**

实现灵活的排序算法，支持按综合成绩或实得学分进行多维度排序（升序 / 降序）。

**1.4人员分工**

**1.5开发计划**

需求分析与设计阶段（6.2）

明确系统功能边界（如文件格式、计算规则、交互流程）

完成数据结构设计和声明（学生、课程类结构）

规划 GUI 界面布局与交互逻辑

核心功能实现阶段（6.3）

数据读写模块（加载 / 保存 A.txt、B.txt）

成绩计算的具体实现（综合成绩 / 实得学分算法）

基础数据结构实现（学生 / 课程管理类）

功能实现与界面开发（6.4）

实现查询、删除、排序功能

开发图形界面

优化对话框交互逻辑

测试与优化阶段（6.5）

单元测试（成绩计算、文件读写）

集成测试（功能联动、数据一致性）

性能优化（查询效率、内存管理）

文档编写与部署（6.6）

撰写实验报告

**1.6开发环境**

开发工具：

Qt Creator IDE (Qt 6.5.3 )：用于辅助完成图形化界面的设计。

C++ 编译器 (VS2022)：用于完成数据结构中类和对象的设计。

开发依赖库：

Qt Widgets (GUI 组件)

Qt Gui (图形处理)

STL (标准模板库)

## 2 总体设计

**2.1 软件的总体结构**

学生管理系统采用模块化设计，基于 C++ 和 Qt框架开发，分为用户界面层、数据管理层和数据存储层三个部分。系统通过主界面导航到六个子界面，子界面调用数据管理模块操作数据文件，实现学生信息和课程成绩的管理。以下是系统总体结构框图（见图 2-1）：

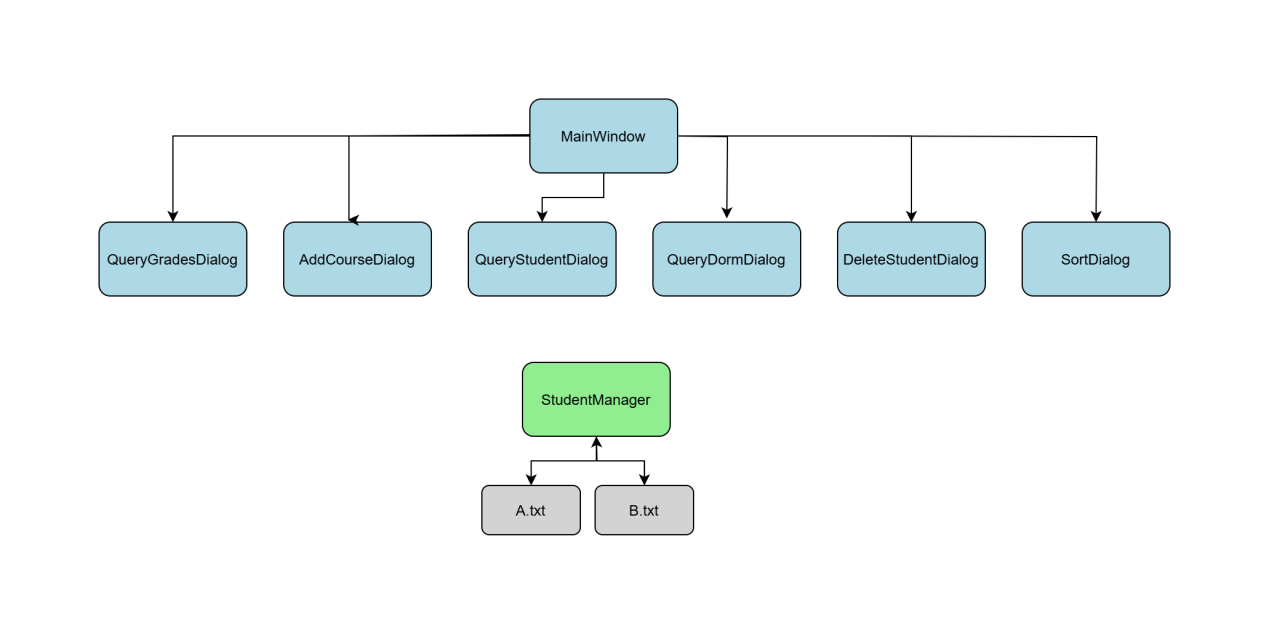


图 2-1：系统总框图

**用户界面层：**

主界面（MainWindow）：系统的入口，显示背景图片和六个导航按钮（录入课程、查询学生、查询宿舍、查询成绩、删除学生、排序成绩），负责跳转到子界面。

子界面：包括六个对话框（AddCourseDialog、QueryStudentDialog、QueryDormDialog、QueryGradesDialog、DeleteStudentDialog、SortDialog），处理用户输入和输出。

**数据管理层（StudentManager）：**

管理学生和课程数据，提供核心功能，如文件读写、添加课程、删除学生、查询信息、排序成绩。

使用 Student 和 Course 类存储数据，QList 管理数据列表。

**数据存储层：**

数据存储在两个文本文件：A.txt（学生信息：学号、姓名、性别、宿舍、电话）和 B.txt（课程信息：学号、课程编号、名称、学分、成绩）。

各层通过函数调用交互：主界面触发子界面，子界面调用 StudentManager 的方法操作数据，数据管理层读写文件，完成功能。

**2.2 模块功能及关系**

系统划分为七个主要模块，分为一个主界面模块、五个子界面模块和一个数据管理模块。以下是各模块的功能及关系：

**数据管理模块（StudentManager）**

**功能：**读取 A.txt 和 B.txt，解析数据到 QList<Student> 和 QList<Course>。

添加课程（addCourse）：将新课程添加到列表并保存到 B.txt。

删除学生（deleteStudent）：移除学生信息及其课程并将结果更新到B.txt。

查询学生（findStudentsByIdOrName、findStudentsByDorm）：按学号、姓名或宿舍查找。

查询成绩（findCoursesByStudentId）：列出学生课程和总学分。

排序成绩（sortCoursesByFinalScore、sortCoursesByActualCredit）：按综合成绩或实得学分排序。

**关系：**为核心模块，接受子界面的请求，操作数据文件，返回结果。

**主界面模块（MainWindow）**

**功能：**加载背景图片（background.jpg），动态调整窗口大小。显示标题和六个导航按钮，点击按钮打开对应子界面。

**关系：**作为系统入口，触发子界面，与数据管理模块无直接交互。

**录入课程模块（AddCourseDialog）**

**功能：**

提供输入框（学号、课程编号、名称、学分、成绩）。

验证输入（如学分 > 0，成绩在 0-100 或 -1），调用 StudentManager::addCourse。

**关系：**依赖 StudentManager 保存数据，返回结果提示。

**查询学生模块（QueryStudentDialog）**

**功能：**

输入学号或姓名，调用 StudentManager::findStudentsByIdOrName。

在 QTextEdit 显示查询结果。

**关系：**依赖 StudentManager 获取数据。

**查询宿舍模块（QueryDormDialog）**

**功能：**

输入宿舍编号，调用 StudentManager::findStudentsByDorm。

显示宿舍学生列表。

**关系：**依赖 StudentManager 获取数据。

**查询成绩模块（QueryGradesDialog）**

**功能：**输入学号，调用 StudentManager::findCoursesByStudentId。显示课程列表、综合成绩和总学分。

**关系：**依赖 StudentManager 获取数据。

**删除学生模块（DeleteStudentDialog）**

**功能：**

输入学号，调用 StudentManager::deleteStudent，更新 B.txt。

显示操作结果。

**关系：**依赖 StudentManager 修改数据。

**排序成绩模块（SortDialog）**

**功能：**

提供四种排序方式（综合成绩/实得学分，升序/降序），调用 StudentManager::sortCoursesByFinalScore 或 sortCoursesByActualCredit。

显示排序结果。

**关系：**依赖 StudentManager 获取排序数据。

**模块关系：**

主界面通过信号和槽机制（如按钮点击事件）触发子界面。

子界面通过 StudentManager 的实例指针调用数据操作方法。

数据管理模块与文件直接交互，统一管理数据，确保一致性。

**2.3 总体流程设计**

系统的总体流程分为初始化、用户操作和数据保存三个阶段。以下是总体流程描述及流程图（见图 2-2）：

**初始化阶段：**

程序启动，MainWindow 构造，加载背景图片，设置窗口大小，初始化按钮和布局。

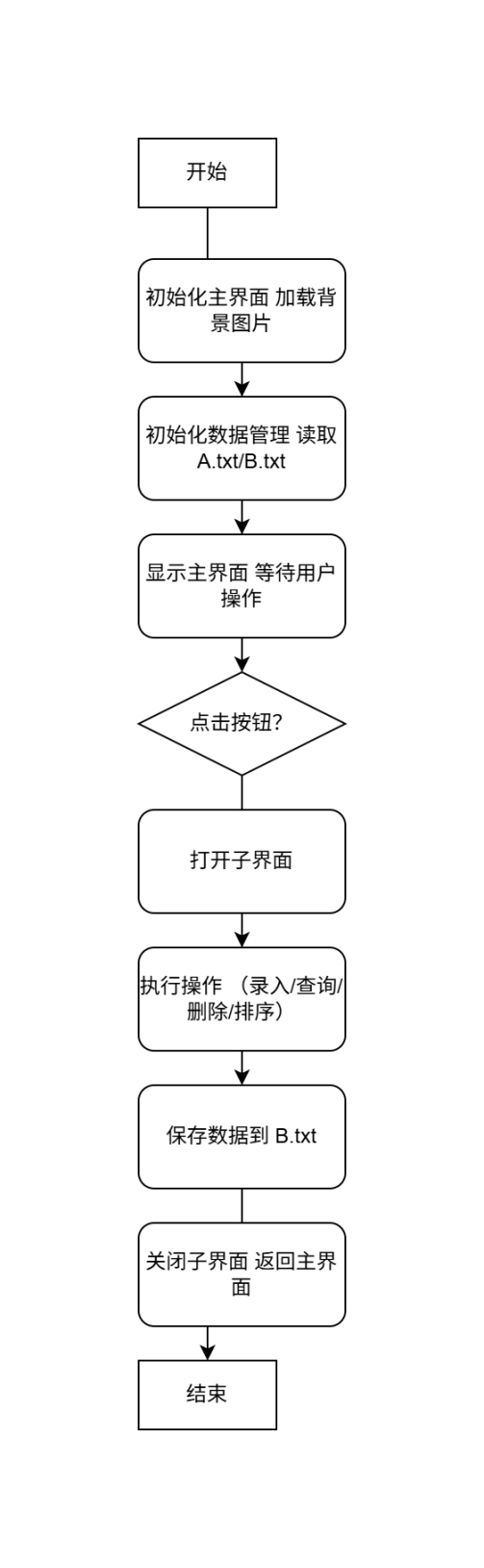
StudentManager 实例化，调用 loadStudents 和 loadCourses，从 A.txt 和 B.txt 读取数据，解析到 QList。

若文件不存在或格式错误，输出错误提示，使用空列表。

**用户操作阶段：**

用户在主界面点击按钮，打开对应子界面（如 AddCourseDialog）。

子界面接受用户输入，验证有效性后调用 StudentManager 的方法：

录入课程：验证输入，调用 addCourse，更新 B.txt。

查询：输入学号、姓名或宿舍，调用查询方法，显示结果。

删除学生：输入学号，调用 deleteStudent，更新 B.txt。

排序成绩：选择排序方式，调用排序方法，显示结果。

操作结果通过 QMessageBox 或 QTextEdit 反馈给用户。

**数据保存阶段：**

添加课程或删除学生后，StudentManager 调用 saveCourses，将课程列表写入 B.txt。

程序退出时，数据自动保存，确保下次启动时数据一致。

**流程特点：**

模块化：各模块功能独立，降低耦合，便于维护。

用户友好：子界面提供输入验证和错误提示，提升体验。

数据一致性：所有数据操作通过 StudentManager，确保文件与内存同步。

图

图2-2：总体流程图

## 3 详细设计

**3.1 数据管理模块**

**3.1.1 模块流程和实现算法**

**功能：**管理学生和课程数据，负责文件读写、添加课程、删除学生、查询信息、排序成绩。

**内部流程：**

程序启动，调用 loadStudents 和 loadCourses，从 A.txt 和 B.txt 读取数据。

使用 QRegularExpression 解析每行，验证字段完整性，创建 Student 或 Course 对象，存储到 QList。

根据子界面请求，执行添加、删除、查询或排序操作：

添加课程：验证输入，创建 Course 对象，追加到 courses 列表，保存到 B.txt。

删除学生：移除学生及其课程，更新 B.txt。

查询：遍历 students 或 courses 列表，返回匹配结果。

排序：使用 std::sort 按综合成绩或实得学分排序。

操作完成后，调用 saveCourses 更新 B.txt。

**实现算法：**

文件解析：逐行读取，使用正则表达式QRegularExpression \\s+ 分割字段，验证字段数量（学生 5 字段，课程 7 字段）。

查询：线性搜索，时间复杂度 O(n)，适合小型文本文件数据。

排序：基于 C++ STL 的 std::sort，时间复杂度 O(n log n)，使用自定义比较函数通过lambda表达式来判断排序的升序或降序。流程图（见图 3-1）：

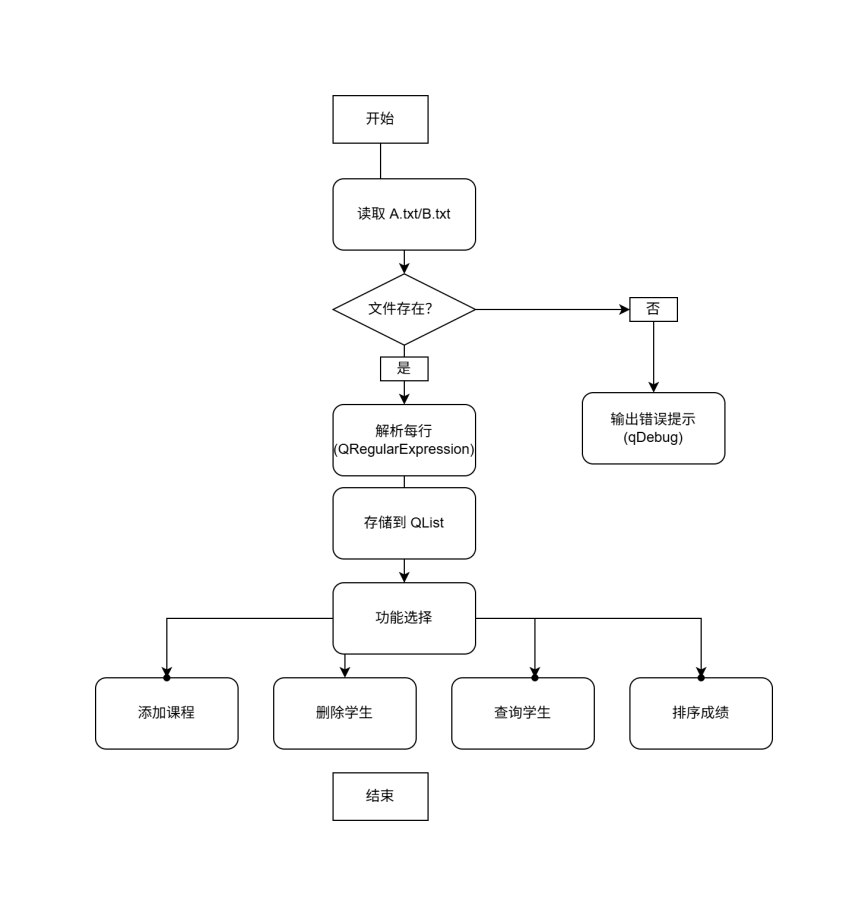


图 3-1：数据管理模块流程图

**3.1.2 类、对象、函数描述**

**类：Student（使用VS2022实现）**

功能：存储学生信息。空实现，Student类仅作为数据容器来使用。

成员变量：

String id：学号，唯一标识。

String name：姓名。

String gender：性别（男/女）。

String dorm：宿舍编号。

String phone：电话号码。

构造函数：Student(const String &id, const String &name, const String &gender,

const String &dorm, const String &phone)

参数：初始化学生信息。

功能：设置成员变量。

**类：Course（使用VS2022实现）**

功能：存储课程信息，计算综合成绩和实得学分。

成员变量：

String studentId：学号，关联学生。

String courseId：课程编号。

String courseName：课程名称。

double regularScore; // 平时成绩

double labScore; // 实验成绩

double examScore; // 卷面成绩

double finalScore; // 综合成绩

double actualCredit; // 实得学分

构造函数：Course(const String&studentId, const String&courseId, const String& courseName,

double credit, double regularScore, double labScore, double examScore)

: studentId(studentId), courseId(courseId), courseName(courseName),

credit(credit), regularScore(regularScore), labScore(labScore),

examScore(examScore) {

calculateFinalScore();

calculateActualCredit();

}

参数：初始化课程信息。

功能：设置成员变量。

成员函数：

void calculateFinalScore()：返回综合成绩（按项目要求进行计算）。

void calculateActualCredit()：返回实得学分（按项目要求进行计算）。

**类：StudentManager（使用VS2022实现）**

功能：管理学生和课程列表，提供核心操作。

成员变量：

QList<Student> students：学生列表。

QList<Course> courses：课程列表。

主要函数：

void loadStudents(const String &filename)

参数：filename（文件路径，如 “A.txt”）。

功能：读取学生信息，解析每行，存入 students。

void loadCourses(const String &filename)

参数：filename（文件路径，如 “B.txt”）。

功能：读取课程信息，存入 courses。

void addCourse(const Course& course);

参数：

course（课程类对象）

功能：创建 course对象，追加到 courses，调用 saveCourses。

void deleteStudent(const String &studentId)

参数：studentId（学号）。

功能：移除学生及其课程成绩，更新 B.txt。

QList<Student> findStudentsByIdOrName(const String& keyword)

功能：返回匹配的学生列表。

QList<Student> findStudentsByDorm(const String &dorm)

参数：dorm（宿舍编号）。

功能：返回宿舍学生列表。

QList<Course> findCoursesByStudentId(const String &studentId)

参数：studentId（学号）。

功能：返回对应学生课程列表。

QList<Course> sortCoursesByFinalScore(bool ascending) const

参数：ascending（true 升序，false 降序）。

功能：按综合成绩排序课程。

QList<Course> sortCoursesByActualCredit(bool ascending) const

参数：ascending（true 升序，false 降序）。

功能：按实得学分排序课程。

void saveCourses(const String &filename)

参数：filename（文件路径，如 “B.txt”）。

功能：将 course 写入文件。

**3.2 主界面模块**

**3.2.1 模块流程和实现算法**

**功能：**提供系统入口，显示背景图片和导航按钮，跳转到子界面。

**内部流程：**

构造 MainWindow，调用 setupUI 初始化界面。

加载 bkImage.jpg为背景图片。

创建标题（“学生管理系统”）和六个按钮，设置居中布局。

用户点击按钮，调用相应的函数（触发槽函数），打开对应子界面（如 AddCourseDialog）。

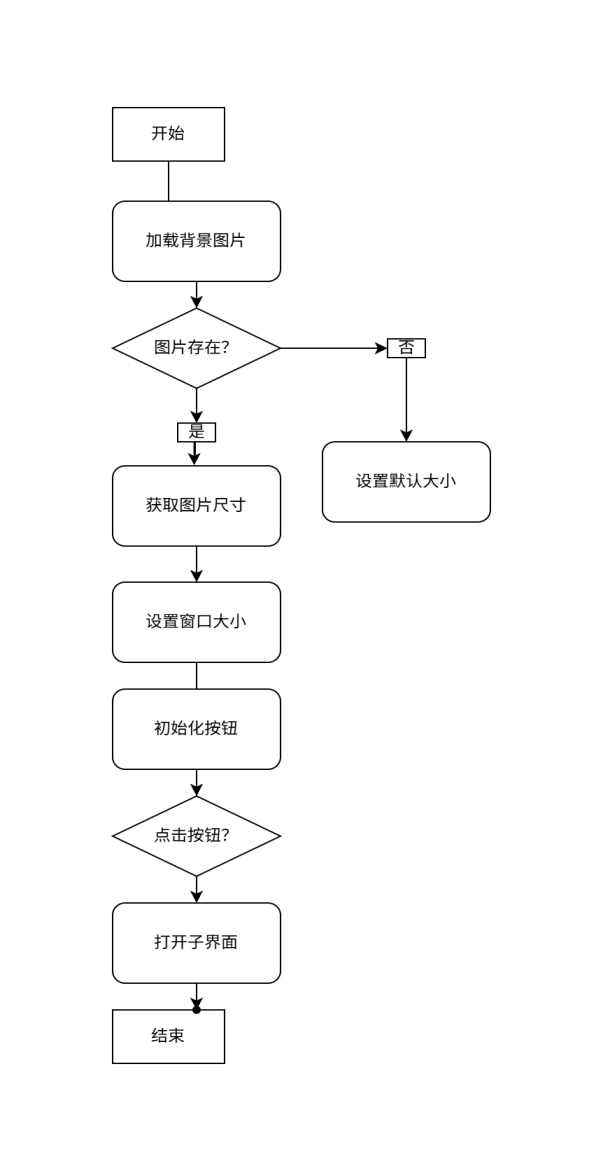
子界面关闭后，返回主界面。

**实现算法：**

图片加载：使用 QPixmap 读取图片，检查 isNull()，动态设置窗口大小。

布局：使用 QVBoxLayout 居中对齐标题和按钮，间距固定。

流程图（见图 3-2）：



**3.2.2 类、对象、函数描述**

**类：MainWindow（使用Qt Creator实现）**

功能：主窗口，继承 QMainWindow，显示导航界面。

成员变量：

StudentManager manager;数据管理实例指针

QPushButton \*addCourseButton;

QPushButton \*queryStudentButton;

QPushButton \*queryDormButton;

QPushButton \*queryGradesButton;

QPushButton \*deleteStudentButton;

QPushButton \*sortButton;

构造函数：MainWindow(QWidget \*parent = nullptr)

参数：parent（父窗口，默认为空）。

功能：初始化 manager，调用 setupUI。

主要函数：

void setupUI()

参数：无。 图 3-2：主界面模块流程图

功能：加载背景图片，设置窗口大小，创建标题和按钮，调用对应的功能函数（连接信号槽）。

void openAddCourseDialog()、void openQueryStudenDialog() 等

参数：无。

功能：槽函数，创建并显示子界面（如 new AddCourseDialog(&manager, this)）。

**3.3 子界面模块 - 录入课程**

**3.3.1 模块流程和实现算法**

功能：录入课程信息，验证后保存。

**内部流程：**

构造 AddCourseDialog，显示输入框（学号、课程编号、名称、学分、成绩）和提交按钮。

用户输入数据，点击提交。

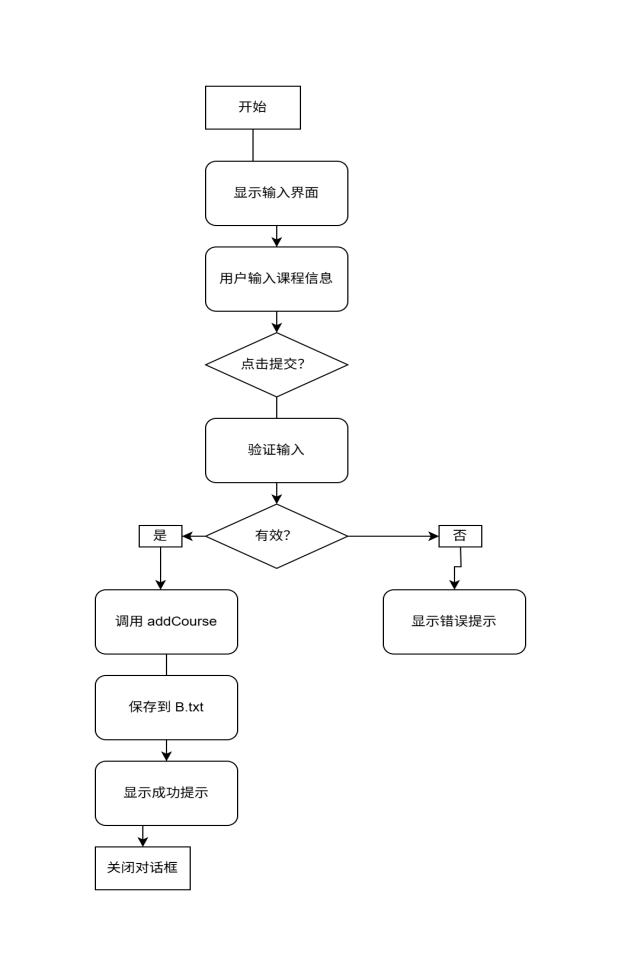
验证输入：

学号、课程编号、名称非空。

学分 > 0。

成绩为 -1 或 0-100。

若有效，调用 StudentManager::addCourse，显示成功提示，关闭对话框；否则显示错误提示。

**实现算法：**

输入验证：使用 QString::isEmpty() 和数值范围检查。

数据保存：委托给 StudentManager::addCourse（封装好的保存函数）。

流程图（见图 3-3）：

**3.3.2 类、对象、函数描述**

**类：AddCourseDialog（使用Qt Creator实现）**

功能：录入课程对话框，继承 QDialog。

成员变量：

StudentManager \*manager：数据管理实例指针。

QLineEdit\*inputStudentId;\*inputCourseId;\*inputCourseName;\*inputCredit;\*inputRegularScore;\*inputLabScore;\*inputExamScore;：输入框。

QPushButton \*submitButton：提交按钮。

图 3-3：录入课程模块流程图

构造函数：AddCourseDialog(StudentManager \*manager, QWidget \*parent = nullptr)

参数：

manager：数据管理实例。

parent：父窗口。

功能：初始化界面，连接提交按钮信号。

主要函数：

void addCourse();

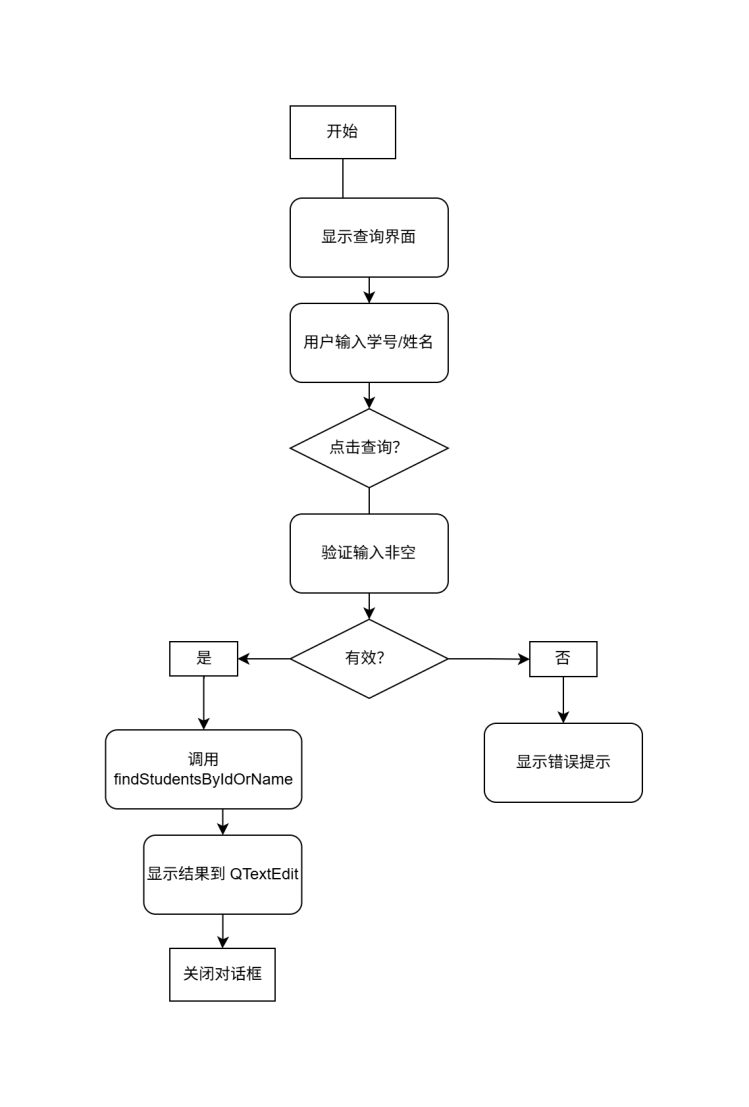
参数：无。

功能：验证输入，调用 manager->addCourse，将添加的课程添加到B.txt中。

**3.4 子界面模块 - 查询学生**

**3.4.1 模块流程和实现算法**

功能：按学号或姓名查询学生信息。

**内部流程：**

构造 QueryStudentDialog，显示输入框（学号/姓名）、查询按钮和结果显示区（QTextEdit）。

用户输入关键字，点击查询，触发 on\_queryButton\_clicked。验证输入非空，调用 StudentManager::findStudentsByIdOrName。

将结果格式化，显示在 QTextEdit；若无结果，显示提示。

**实现算法：**

查询：StudentManager，返回 QList<Student>，逐项格式化。流程图（见图 3-4）：

图 3-4：查询学生模块流程图

**3.4.2 类、对象、函数描述**

**类：QueryStudentDialog（使用Qt Creator实现）**

功能：查询学生对话框，继承 QDialog。

成员变量：

StudentManager \*manager：数据管理实例。

QLineEdit \*keywordEdit：输入框。

QTextEdit \*resultTextEdit：结果显示区。

QPushButton \*queryButton：查询按钮。

构造函数：QueryStudentDialog(StudentManager \*manager, QWidget \*parent = nullptr)

参数：manager, parent。

功能：初始化界面，连接查询按钮信号。

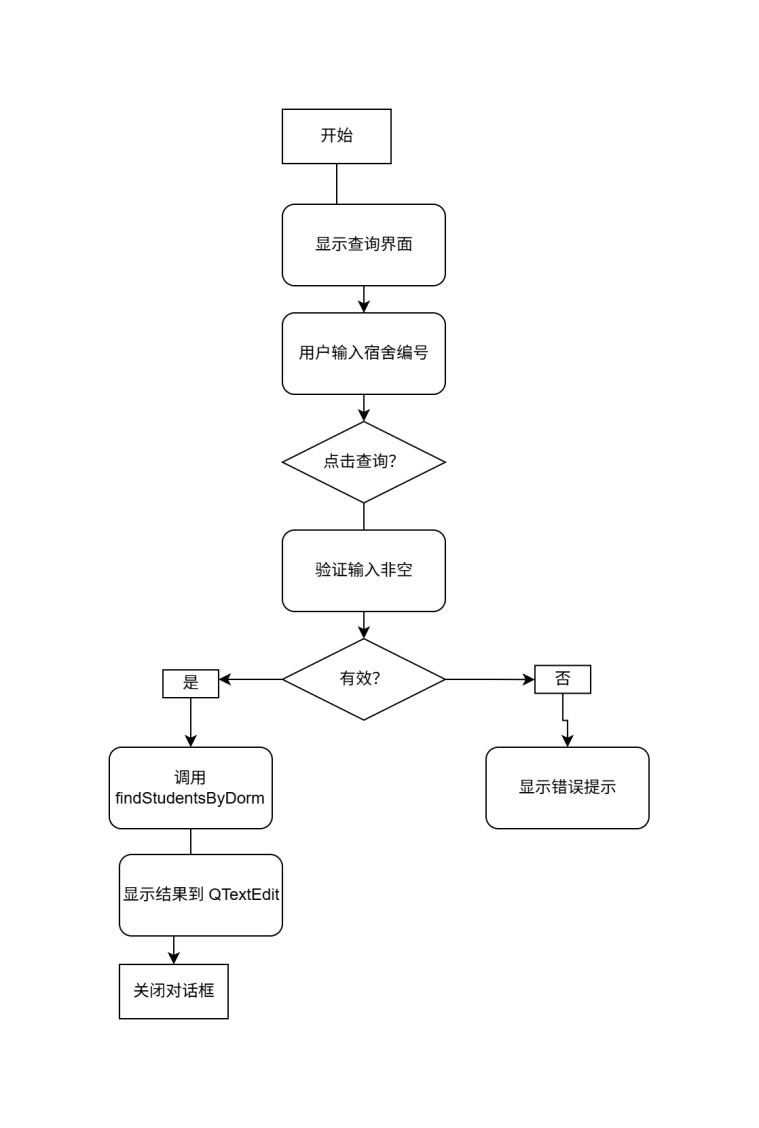
主要函数：

void queryStudent();

参数：无。

功能：验证输入，调用 manager->findStudentsByIdOrName，显示结果。

**3.5 子界面模块 - 查询宿舍**

**3.5.1 模块流程和实现算法**

功能：按宿舍编号查询学生列表。

**内部流程：**

同查询学生模块，输入宿舍编号，调用 StudentManager::findStudentsByDorm，显示结果。

**实现算法：**

同查询学生，使用 StudentManager 提供的线性搜索。

流程图（见图 3-5）：

**3.5.2 类、对象、函数描述**

**类：QueryDormDialog（使用Qt Creator实现）**

功能：查询宿舍对话框，继承 QDialog。

成员变量：同 QueryStudentDialog，输入框为宿舍编号。

构造函数：同 QueryStudentDialog。

主要函数：

void queryDorm();

参数：无。

功能：调用 manager->findStudentsByDorm，显示结果。

图 3-5：查询宿舍模块流程图

**3.6 子界面模块 - 查询成绩**

**3.6.1 模块流程和实现算法**

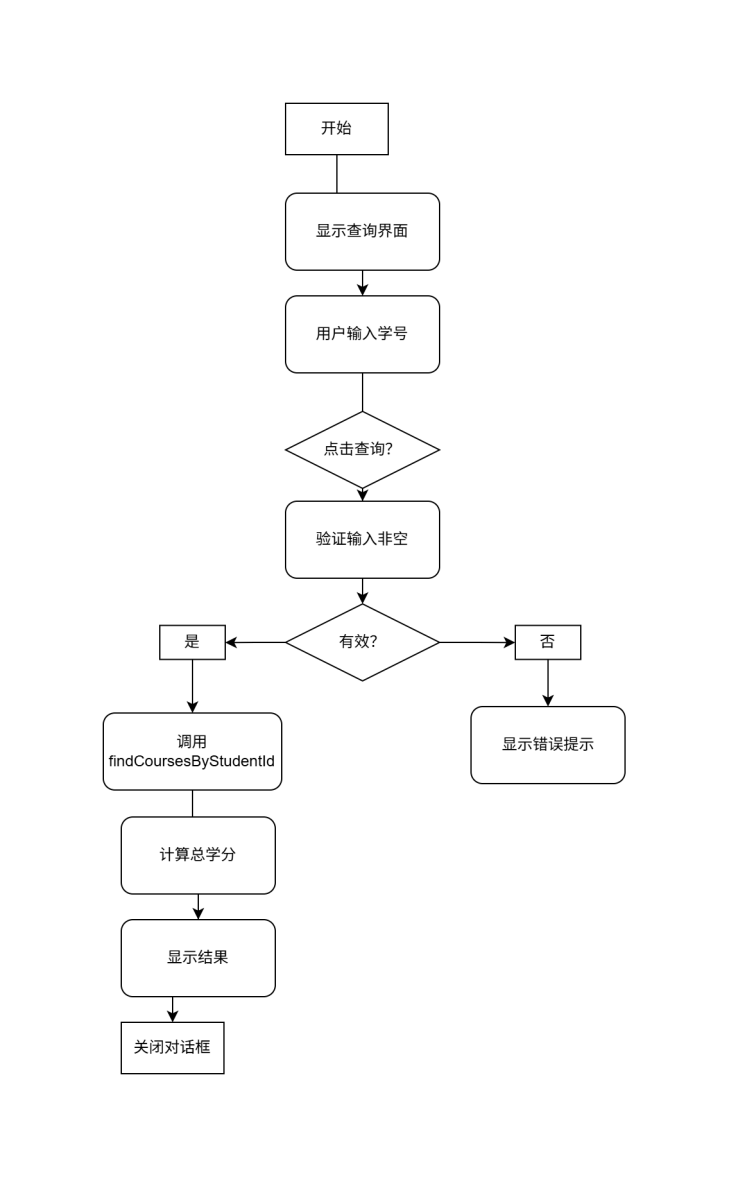
功能：按学号查询课程成绩和总学分。

**内部流程：**

构造 QueryGradesDialog，显示输入框（学号）、查询按钮和结果显示区。

用户输入学号，点击查询，验证非空。

调用 StudentManager::findCoursesByStudentId，计算总学分，显示结果。



**实现算法：**

总学分计算：遍历课程，累加 calculateActualCredit()。

流程图（见图 3-6）：

**3.6.2 类、对象、函数描述**

**类：QueryGradesDialog（使用Qt Creator实现）**

功能：查询成绩对话框，继承 QDialog。

成员变量：同 QueryStudentDialog，输入框为学号。

构造函数：同 QueryStudentDialog。

主要函数：

void querygrades()

参数：无。

功能：调用 manager->findCoursesByStudentId，计算总学分，显示结果。

图 3-6：查询成绩模块流程图

**3.7 子界面模块 - 删除学生**

**3.7.1 模块流程和实现算法**

功能：删除学生及其课程。

流程：

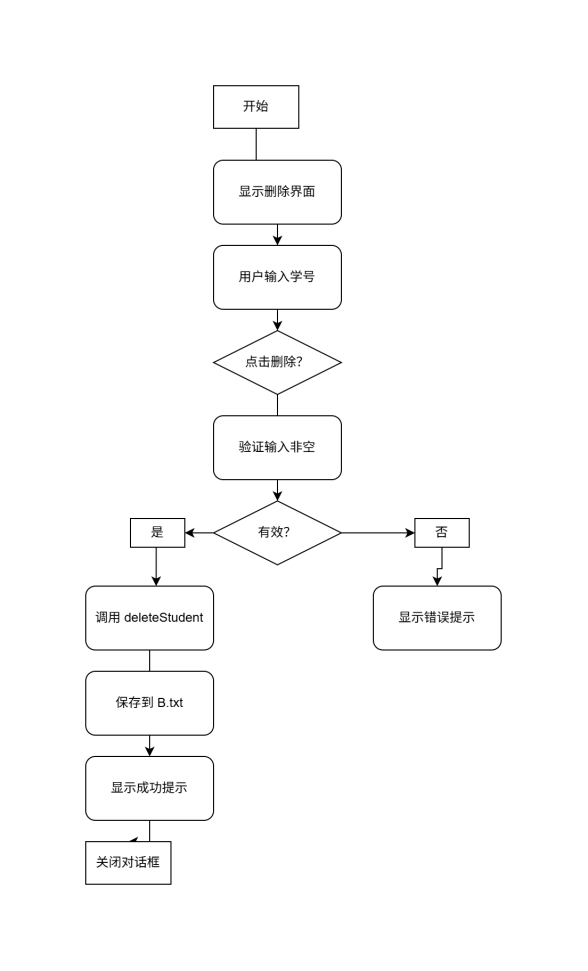
构造 DeleteStudentDialog，显示输入框（学号）和删除按钮。

用户输入学号，点击删除，验证非空。

调用 StudentManager::deleteStudent，显示结果，关闭对话框。

算法：委托给 StudentManager，移除学生和课程。

流程图（见图 3-7）：

**3.7.2 类、对象、函数描述**

**类：DeleteStudentDialog（使用Qt Creator实现）**

功能：删除学生对话框，继承 QDialog。

成员变量：同 QueryStudentDialog，输入框为学号。

构造函数：同 QueryStudentDialog。

主要函数：

void deleteStudent()

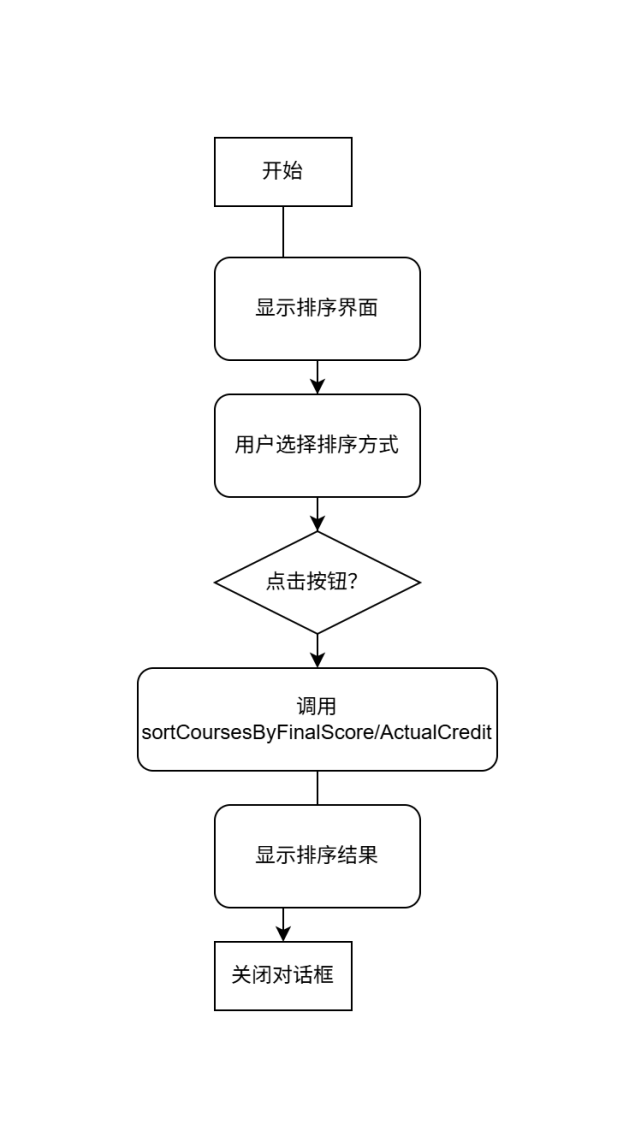
参数：无。

功能：调用 manager->deleteStudent，显示提示。

**3.8 子界面模块 - 排序成绩**

**3.8.1 模块流程和实现**

功能：按综合成绩或实得学分排序课程。 图 3-7：删除学生模块流程图

流程：

构造 SortDialog，显示排序选项（综合成绩/实得学分，升序/降序）和排序按钮。

用户选择排序方式，点击按钮，触发 on\_sortButton\_clicked。

调用 StudentManager::sortCoursesByFinalScore 或 sortCoursesByActualCredit，显示结果。

算法：使用 std::sort，时间复杂度 O(n log n)。

流程图（见图 3-8）：

**3.8.2 类、对象、函数描述**

**类：SortDialog（使用Qt Creator实现）**

功能：排序成绩对话框，继承 QDialog。

成员变量：

StudentManager \*manager：数据管理实例。

QComboBox \*sortTypeCombo, \*orderCombo：排序选项。

QTextEdit \*resultTextEdit：结果显示区。 图 3-8：排序成绩模块流程图

QPushButton \*sortButton：排序按钮。

构造函数：SortDialog(StudentManager \*manager, QWidget \*parent = nullptr)

参数：manager, parent。

功能：初始化界面，设置选项。

主要函数：

void sortByFinalScoreAsc();

void sortByFinalScoreDesc();

void sortByActualCreditAsc();

void sortByActualCreditDesc();

参数：无。

功能：根据选项调用排序（升序或降序）函数，显示结果。

## 

## 4 编码设计

**4.1 开发环境的设置和建立**

开发环境已在报告中详细描述（见开发环境章节），以下简要总结设置步骤：

开发工具：

Qt Creator 10.0.2：IDE，用于代码编辑、调试和界面设计。

MinGW 64-bit (8.1.0)：编译器。

Qt 6.5.3：框架，提供 GUI 和核心功能。

项目配置：

路径：D:\桌面\qt\CourseDesign\test。

项目文件

test.pro：QT += core gui widgets

CONFIG += c++17

SOURCES += main.cpp mainwindow.cpp studentmanager.cpp addcoursedialog.cpp ...

HEADERS += mainwindow.h studentmanager.h addcoursedialog.h ...

RESOURCES += resources.qrc

数据文件（A.txt, B.txt）和背景图片（background.jpg）放置于 build/debug/。

设置步骤：

安装 Qt 和 Qt Creator，选择 Qt 6.5.3 和 MinGW 组件。

在 Qt Creator 中创建 Qt Widgets 项目，配置 MinGW 编译器。

添加源文件、头文件和资源，设置 C++17 标准。

复制 A.txt, B.txt, bkImage.jpg 到运行目录。

编译运行，检查环境配置。

**4.2 程序设计时要注意的事项**

**代码规范：**

使用驼峰命名（如 StudentManager, addCourse），变量小写首字母（如 studentId）。

每函数不超过 50 行，保持单一职责。

添加详细注释，说明函数功能、参数和返回值。

**内存管理：**

Qt 对象的父子关系自动管理内存（如 QPushButton 设置父对象），但需注意手动 new 的对象（如 StudentManager）在程序结束时删除。

避免内存泄漏，使用 delete释放在堆区建立的动态内存。

**文件操作：**

确保 A.txt 和 B.txt 格式正确（每行字段用空格分隔）。

检查文件是否存在，添加错误提示（如 QMessageBox）。

**输入验证：**

子界面输入需验证（如学分 > 0，成绩在 -1 或 0-100 范围）。

使用 QString::isEmpty() 和 toDouble() 检查输入有效性。

**4.3 关键构件/插件的特点和使用**

项目依赖 Qt 框架的关键构件，以下是主要构件特点和使用：

QMainWindow：

特点：提供主窗口框架，支持菜单栏、工具栏和中心部件。

使用：MainWindow 继承 QMainWindow，设置中心部件 QWidget，包含标题和按钮。

代码示例：setCentralWidget(new QWidget(this));

QDialog：

特点：模态对话框，适合用户交互（如输入、查询）。

使用：子界面（如 AddCourseDialog）继承 QDialog，显示输入框和按钮。

代码示例：AddCourseDialog dialog(manager, this);dialog.exec();

QVBoxLayout：

特点：垂直布局，自动调整部件位置，支持居中对齐。

使用：主界面和子界面使用 QVBoxLayout 排列标题、输入框和按钮。

代码示例：QVBoxLayout \*layout = new QVBoxLayout(centralWidget);

layout->setAlignment(Qt::AlignCenter);

QRegularExpression：

特点：支持正则表达式，高效解析文本。

使用：StudentManager 使用 QRegularExpression 解析 A.txt 和 B.txt。

代码示例：QRegularExpression re("\\s+");

QStringList parts = line.split(QRegularExpression("\\s+"));

QMessageBox：

特点：弹出提示框，显示成功或错误信息。

使用：子界面验证输入失败时显示警告。

代码示例：QMessageBox::warning(this, "错误", "输入无效！");

**4.4 主要程序的代码设计及注释**

以下是三个关键模块的代表性代码片段，包含详细注释。

**4.4.1 数据管理模块（StudentManager::addCourse）**

// 添加课程到课程列表并保存到文件

void StudentManager::addCourse(const QString &studentId, const QString &courseId,

const QString &courseName, double credit, double score) {

// 创建 Course 对象，存储课程信息

courses.append(Course(studentId, courseId, courseName, credit, score));

// 保存课程列表到 B.txt

saveCourses("B.txt");

}

// 将课程列表写入文件

void StudentManager::saveCourses(const QString &filename) {

QFile file(filename);

// 以写入模式打开文件

if (!file.open(QIODevice::WriteOnly | QIODevice::Text)) {

qDebug() << "无法打开文件:" << filename;

return;

}

QTextStream out(&file);

// 遍历课程列表，写入每门课程信息

for (const auto &course : courses) {

out << course.studentId() << " " << course.courseId() << " "

<< course.courseName() << " " << course.credit() << " "

<< course.score() << "\n";

}

file.close(); // 关闭文件

}

说明：addCourse 负责添加课程，调用 saveCourses 更新 B.txt，确保数据持久化。

**4.4.2 主界面模块（MainWindow::setupUI）**

// 初始化主界面布局和背景

void MainWindow::setupUI() {

// 创建中心部件

QWidget \*centralWidget = new QWidget(this);

setCentralWidget(centralWidget);

// 加载背景图片

QPalette palette;

QPixmap background("background.jpg");

if (!background.isNull()) {

// 若图片有效，调整窗口大小并设置背景

palette.setBrush(QPalette::Window, background.scaled(size()));

centralWidget->setPalette(palette);

centralWidget->setAutoFillBackground(true);

resize(background.size());

} else {

// 若图片加载失败，使用默认大小

resize(800, 600);

}

// 创建垂直布局，居中对齐

QVBoxLayout \*mainLayout = new QVBoxLayout(centralWidget);

mainLayout->setAlignment(Qt::AlignCenter);

// 添加标题

QLabel \*title = new QLabel("学生管理系统", centralWidget);

title->setAlignment(Qt::AlignCenter);

mainLayout->addWidget(title);

// 添加导航按钮

addCourseButton = new QPushButton("录入课程信息", this);

queryStudentButton = new QPushButton("查询学生信息", this);

queryDormButton = new QPushButton("查询宿舍学生", this);

queryGradesButton = new QPushButton("查询学生成绩", this);

deleteStudentButton = new QPushButton("删除学生信息", this);

sortButton = new QPushButton("排序成绩信息", this);

// 按钮样式（进行统一设置）

QString buttonStyle = "QPushButton { font-size: 16px; padding: 10px; background-color: #1011A1; color: white; border-radius: 5px; }"

"QPushButton:hover { background-color: #45a049; }";

addCourseButton->setStyleSheet(buttonStyle);

queryStudentButton->setStyleSheet(buttonStyle);

queryDormButton->setStyleSheet(buttonStyle);

queryGradesButton->setStyleSheet(buttonStyle);

deleteStudentButton->setStyleSheet(buttonStyle);

sortButton->setStyleSheet(buttonStyle);

mainLayout->addWidget(addCourseButton);

mainLayout->addWidget(queryStudentButton);

mainLayout->addWidget(queryDormButton);

mainLayout->addWidget(queryGradesButton);

mainLayout->addWidget(deleteStudentButton);

mainLayout->addWidget(sortButton);

mainLayout->addStretch();

// 连接信号和槽

//利用Qt中的信号与槽机制（实现组件间的通信）

//slot：接收信号后需要调用的函数

connect(addCourseButton,&QPushButton::clicked,this,&MainWindow::openAddCourseDialog);

connect(queryStudentButton,&QPushButton::clicked,this,&MainWindow::openQueryStudentDialog);

connect(queryDormButton,&QPushButton::clicked,this,&MainWindow::openQueryDormDialog);

connect(queryGradesButton,&QPushButton::clicked,this,&MainWindow::openQueryGradesDialog);

connect(deleteStudentButton,&QPushButton::clicked,this,&MainWindow::openDeleteStudentDialog);

connect(sortButton,&QPushButton::clicked,this, &MainWindow::openSortDialog);

}

}

}

说明：setupUI 初始化主界面，加载背景图片，设置居中布局，动态连接按钮信号。

**4.4.3 子界面模块（DeleteStudentDialog::DeleteStudentDialog）**

// 处理提交按钮点击事件，验证并保存

DeleteStudentDialog::DeleteStudentDialog(StudentManager \*manager, QWidget \*parent)

: QDialog(parent), manager(manager) {

setWindowTitle("删除学生信息");

QVBoxLayout \*mainLayout = new QVBoxLayout(this);

inputStudentId = new QLineEdit(this);

inputStudentId->setPlaceholderText("输入学号 (如: 01)");

mainLayout->addWidget(new QLabel("学号:", this));

mainLayout->addWidget(inputStudentId);

submitButton = new QPushButton("删除", this);

submitButton->setStyleSheet("font-size: 14px; padding: 8px; background-color: #f44336; color: white; border-radius: 5px;");

mainLayout->addWidget(submitButton);

connect(submitButton,&QPushButton::clicked,this,&DeleteStudentDialog::deleteStudent);

}

// 调用 StudentManager 中封装好的删除函数将处理的结果保存至B.txt中

manager->deleteStudent(studentId);

// 显示成功提示QMessageBox::information(this, "成功", "学生删除成功！");

accept();// 关闭对话框

}

说明：on\_submitButton\_clicked 验证用户输入，调用 addCourse 保存数据，提供反馈。

**4.5 解决的技术难点及经常犯的错误**

**4.5.1 技术难点**

**文件解析与格式验证：**

问题：A.txt 和 B.txt 可能包含多余空格或格式错误，导致解析失败。

解决：使用 QRegularExpression("\\s+") 分割字段，验证字段数量（学生 5 字段，课程 5 字段），添加错误提示。

代码：QStringList parts = line.split(QRegularExpression("\\s+"), Qt::SkipEmptyParts);

if (parts.size() >= 5) { /\* 解析数据 \*/ }

**背景图片加载失败：**

问题：background.jpg 路径错误或文件缺失，导致主界面空白。

解决：检查 QPixmap::isNull()，设置默认窗口大小，确保图片位于 build/debug/。

代码：if (background.isNull()) { resize(600, 400); }

**输入验证复杂性：**

问题：用户输入无效数据（如负学分），导致程序崩溃。

解决：在子界面验证输入，使用 QMessageBox 提示错误，确保数据符合要求。

代码：见 AddCourseDialog::on\_submitButton\_clicked。

**4.5.2 常见错误**

**文件路径错误：**

错误：A.txt 或 background.jpg 未放在 build/debug/，导致文件未找到。

纠正：明确文档中说明文件路径，测试前检查目录。

预防：在代码中添加路径检查和错误日志。

**信号槽连接遗漏：**

错误：按钮未连接槽函数，点击无反应。

纠正：使用 connect 确保信号绑定，调试时检查槽函数触发。

预防：在 setupUI 中统一连接信号。

**内存泄漏：**

错误：手动 new 的对象未删除（如 StudentManager）。

纠正：在 MainWindow 析构函数中删除 manager。

预防：利用 Qt 父子关系或智能指针管理内存。

**4.6 特色算法示例**

以下是两个具有代表性的算法，展示实现细节和优化思路。

**4.6.1 算法 1：课程成绩排序（sortCoursesByFinalScore）**

功能：按综合成绩排序课程，支持升序和降序。

实现：

使用 C++ STL 的 std::sort，基于 Course::calculateFinalScore 比较。

时间复杂度：O(n log n)，适合小型数据集。

空间复杂度：O(1)，原地排序。

代码：

// 按综合成绩排序课程

// 通过lambda表达式定义排序的比较逻辑

QList<Course> StudentManager::sortCoursesByFinalScore(bool ascending) {

QList<Course> sortedCourses = courses;

std::sort(sortedCourses.begin(), sortedCourses.end(),

[ascending](const Course& a, const Course& b) {

return ascending?a.finalScore<b.finalScore : a.finalScore >b.finalScore;

});

return sortedCourses;

}

// 计算综合成绩

double Course::calculateFinalScore() const {

if (score == -1) return -1; // 未录入成绩

return score; // 综合成绩（假设为输入成绩，简化示例）

}

特点：

灵活性：支持升序/降序，通过bool类型的ascending变量用于决定排序是升序还是降序。

效率高：利用lambda表达式和std::sort排序函数相结合实现快速排序，性能较为优越。

使用场景：SortDialog 中显示排序结果，满足用户按成绩排序的需求。

**4.6.2 算法 2：学生查询（findStudentsByIdOrName）**

功能：按学号或姓名查询学生。

实现：

使用线性搜索，遍历 students 列表，检查 id 或 name 是否包含关键字。

时间复杂度：O(n)，适合小型数据集。

空间复杂度：O(1)，仅存储结果列表。

代码：

// 按学号或姓名查询学生

QList<Student> StudentManager::findStudentsByIdOrName(const QString& keyword) {

QList<Student> result;

//Based-for循环：常用于遍历容器中的元素，没有显式的索引，更加高效

//遍历容器找到匹配的所有值，存储在Student容器中

for (const Student& student : students) {

if (student.id == keyword || student.name == keyword) {

result.append(student);

}

}

return result;

}

特点：

简单高效：based-for循环遍历容器更加高效便捷线性搜索适合小型数据，代码易维护。

使用场景：QueryStudentDialog 中根据用户输入显示匹配学生。

## 5 测试时出现过的问题及其解决方法

**5.1 文件读写问题**

**问题：**读取 A.txt 或 B.txt 时，若文件包含多余空格或空行，导致解析失败，程序崩溃。

**原因：**StudentManager::loadStudents 和 loadCourses 未充分验证输入格式。

**解决方法：**

使用 QRegularExpression("\\s+") 分割字段，忽略多余空格。

添加字段数量检查（学生5字段，课程7字段），跳过空行或格式错误的行。

**示例代码：**QStringList parts = line.split(QRegularExpression("\\s+"));

if (parts.size() >= 5) { /\* 解析数据 \*/ } else { continue; }

**效果：**程序能正确解析不规范文件，提高了容错性。

**5.2 背景图片加载失败**

**问题：**主界面背景图片（bkImage.jpg）未加载，窗口显示空白或默认灰色。

原因：图片未放置在 build/debug/ 目录，或路径错误。

**解决方法：**

通过检查 QPixmap::isNull()，若图片无效则使用默认窗口大小（600x400）。

在文档中明确图片路径要求，测试前复制文件到运行目录。

**示例代码：**QPixmap background("background.jpg");

if (background.isNull()) { resize(600, 400); }

**效果：**主界面始终显示正常，增强了用户体验。

**5.3 输入验证不足**

**问题：**在 AddCourseDialog 中输入无效数据（如负学分或空学号），未提示错误，直接保存导致数据异常。

**原因：**输入验证逻辑不完整。

**解决方法：**

加强验证，检查学号、课程名称非空，学分 > 0，成绩为 -1 或 0-100。

使用 QMessageBox 提示用户错误，阻止无效操作。

示例代码：if (studentId.isEmpty() || credit <= 0 || (score < 0 && score != -1) || score > 100) {

QMessageBox::warning(this, "错误", "输入无效！");

return;

}

**效果：**用户输入更安全，数据一致性得到保证。

**5.4 查询结果为空**

**问题：**在 QueryStudentDialog 或 QueryDormDialog 中查询不存在的数据（如无效学号），结果显示区为空，用户体验不佳。

原因：未处理空结果情况。

**解决方法：**

检查 StudentManager::findStudentsByIdOrName 等函数的返回值，若结果为空，显示“无记录”提示。

**示例代码：**QList<Student> results = manager->findStudentsByIdOrName(keyword);

if (results.isEmpty()) {

resultTextEdit->setText("无记录");

}

**效果：**查询界面更友好，用户能明确查询结果。

**5.5 删除学生后文件未更新**

**问题：**在 DeleteStudentDialog 中删除学生后，B.txt 未同步更新，导致下次启动时数据不一致。

**原因：**StudentManager::deleteStudent 未调用 saveCourses。

**解决方法：**

在删除操作后立即调用 saveCourses，更新 B.txt。

**示例代码：**void StudentManager::deleteStudent(const QString &studentId) {

students.removeIf([&](const Student &s) { return s.getId() == studentId; });

courses.removeIf([&](const Course &c) { return c.studentId() == studentId; });

saveCourses("B.txt");

}

**效果：**数据文件与内存保持一致，避免了数据丢失。

## 6. 总结

**6.1 课程设计的完成情况**

**已完成部分：**

**功能实现：**

录入课程信息（学号、课程编号、名称、学分、成绩）。

查询学生信息（按学号/姓名）、宿舍学生、课程成绩及总学分。

删除学生及其课程。

按综合成绩或实得学分排序（支持升序/降序）。

**界面设计：**

主界面（MainWindow）包含背景图片和六个导航按钮。

六个子界面（AddCourseDialog 等）支持用户交互，布局清晰。

**数据管理：**

使用 StudentManager 管理学生和课程数据，读写 A.txt 和 B.txt。

支持文件解析、数据验证和持久化。

**测试：**

通过单元测试（验证 addCourse, findStudentsByIdOrName 等函数）。

集成测试（主界面跳转、子界面交互）。

系统测试（功能完整性、用户体验）。

**未完成部分（拓展要求）：**

数据库支持：未使用数据库（如 SQLite）存储数据，仅依赖文本文件。

原因：课程要求以文件操作为主，数据库超出初学者能力范围。

改进计划：未来可集成 SQLite，提升数据管理效率。

多语言支持：界面仅支持中文，未实现国际化。

原因：项目规模有限，专注核心功能。

改进计划：使用 Qt 的 tr() 函数添加多语言支持。

性能优化：对于大规模数据（如上万学生），线性搜索和文件读写效率较低。

原因：测试数据规模小（< 1000 条），当前算法足够。

改进计划：引入索引或哈希表优化查询。

**6.2 创新功能**

**动态背景适配：**

主界面根据 background.jpg 的尺寸自动调整窗口大小，增强美观性和灵活性。

代码：QPixmap background("background.jpg");

if (!background.isNull()) { resize(background.size()); }

**错误提示友好：**

子界面使用 QMessageBox 提供详细错误信息（如“学分必须大于 0”），便于用户修正输入。

**6.3 小组成员分工和成绩自我评定**

| **成员** | **任务** | **贡献内容** | **工作量** | **自评成绩** |
| --- | --- | --- | --- | --- |
| **陈康瑜** | **数据结构与文件操作** | **实现Student,Course, StudentManager，文件读写，模块文档** | **30%** | **90-95** |
| **张棕源** | **主界面与背景图片** | **实现 MainWindow，背景图片适配，模块文档** | **20%** | **85-90** |
| **王昆** | **子界面（录入、查询学生、查询宿舍）** | **实现AddCourseDialog, QueryStudentDialog, QueryDormDialog，模块文档** | **25%** | **85-90** |
| **刘峻辰** | **子界面（成绩、删除、排序）与总体文档** | **实现 QueryGradesDialog, DeleteStudentDialog, SortDialog，整合文档** | **25%** | **85-90** |

**6.4 收获、经验、教训和感受**

**收获：**

掌握了 C++ 的类设计、文件操作和 STL 算法（如 std::sort）。

学习了 Qt 框架的基础知识，包括界面布局、信号槽和对话框开发。

提升了团队协作能力，学会了分工、沟通和文档整合。

**经验：**

模块化设计降低代码耦合，便于调试和维护。

提前测试文件格式和路径问题，可避免运行时错误。

**教训：**

初期未充分验证输入，导致测试阶段发现多处问题，增加了修改成本。

文件路径依赖（如 bkImage.jpg）未统一规范，测试时需手动调整。

统筹协调每个人的不同部分的代码难度较大，代码管理效率较低。

**感受：**

项目开发过程充满挑战，从零开始实现一个完整系统让我们充满成就感。

团队协作中，每个人的贡献都不可或缺，沟通和相互支持是成功的关键。

未来希望学习更多高级技术（如数据库、跨平台开发），提升开发能力。

**6.5 未来改进方向（拓展）**

集成数据库（如 SQLite）替换文本文件，提高数据管理效率。

优化查询算法，使用哈希表或索引支持大规模数据。

添加用户权限管理，区分管理员和普通用户。

支持多语言界面。

## 7.附录

**7.1参考文献**

1.[快速入门QT大法\_qt教程菜鸟-CSDN博客](https://blog.csdn.net/qq_44891434/article/details/131615455?ops_request_misc=&request_id=&biz_id=102&utm_term=qt%E6%95%99%E7%A8%8B&utm_medium=distribute.pc_search_result.none-task-blog-2~all~sobaiduweb~default-0-131615455.nonecase&spm=1018.2226.3001.4187)

2.[C++ Qt开发入门到精通全套教程（C语言/C++/Quick/OpenCV/OpenCL/MySQL数据库/设计模式/Qt实战项目）\_哔哩哔哩\_bilibili](https://www.bilibili.com/video/BV1dd7nzjEsm/?spm_id_from=333.337.search-card.all.click)

**7.2源代码**

[YSU\_C-CourseDesign/test at main · kychen-bit/YSU\_C-CourseDesign](https://github.com/kychen-bit/YSU_C-CourseDesign/tree/main/test)