

《C++程序设计》综合设计报告

题目：学生成绩管理系统

学 院\_\_环境与测绘学院\_\_\_\_\_

班 级\_\_遥感2班\_\_\_\_\_\_\_\_\_\_

姓 名\_\_李志宇\_\_\_\_\_\_\_\_\_\_\_

学 号\_\_07234901\_\_\_\_\_\_\_\_\_\_

中国矿业大学

2025年03月

课程设计报告格式要求：

1.报告中应包含一、二级标题；

2.正文宋体小4号，1.5倍行距；

3.报告正文不少于1000字；

4.上交报告电子版，由班级学习委员收齐后打包成压缩包

5.邮件标题：C++公共课-[你的专业和班级]-[实际报告份数]-[班级总人数].zip

6.压缩包内每个文件必须为PDF，且文件名字是：C++公共课-[你的学号]-[你的姓名].pdf。自己命名好文件发给学委。

7.学委发给我的邮件的截止时间为6月6日（周五），晚上22点整。

**8.个人不要发给我你的报告。因为我今年有350多名学生，抱歉无法处理个人报告的提交，太多了。**

课程设计报告评分要点：

1. 报告中C++的用户界面设计情况（20%）
2. 报告中综合设计的系统功能有效及完整性（40%）
3. 报告中的个人总结及体会情况（20%）
4. 报告格式规范程度（20%）

评语：

评分： 日期：

目录

[1. 系统概述 4](#_Toc200125874)

[2. 类设计 4](#_Toc200125875)

[2.1 ChengJiMember（成绩成员基类） 4](#_Toc200125876)

[2.2 PuTongStudent（普通学生类） 4](#_Toc200125877)

[2.3 PangTingStudent（旁听学生类） 5](#_Toc200125878)

[2.4 ChengJiGuanLi（成绩管理类） 5](#_Toc200125879)

[3. 功能设计 5](#_Toc200125880)

[3.1 学生管理 5](#_Toc200125881)

[3.2 科目管理 6](#_Toc200125882)

[3.3 数据展示 6](#_Toc200125883)

[4. 用户界面 6](#_Toc200125884)

[5. 测试模式 7](#_Toc200125885)

[6. 关键技术 7](#_Toc200125886)

[7. 测试用例 8](#_Toc200125887)

[7.1.添加新学生 8](#_Toc200125888)

[7.2.修改成绩 8](#_Toc200125889)

[7.3.按科目排序展示 8](#_Toc200125890)

[7.4.添加新科目 9](#_Toc200125891)

[7.5.删除科目 9](#_Toc200125892)

[7.6.删除学生 9](#_Toc200125893)

[7.7.查询学生 9](#_Toc200125894)

[7.8.错误处理 9](#_Toc200125895)

[7.9.退出系统 10](#_Toc200125896)

[8.体会与总结 10](#_Toc200125897)

## 1. 系统概述

本系统是一个面向对象的学生成绩管理系统，支持管理普通学生和旁听学生的成绩信息。系统提供完整的CRUD功能（创建、读取、更新、删除），支持多种排序方式，并具有直观的表格展示界面。

## 2. 类设计

### 2.1 ChengJiMember（成绩成员基类）

* **职责**：所有学生类型的基类
* **成员变量**：
  + xingming：学生姓名
  + xuehao：学生学号
* **核心方法**：
  + xianshi()：显示学生信息（纯虚函数）
  + getKemuScore()：获取科目成绩
  + xiugaiScore()：修改科目成绩
  + addSubject()/removeSubject()：添加/删除科目
* **特性**：
  + 重载<运算符实现按学号排序
  + 支持多态（通过虚函数）

### 2.2 PuTongStudent（普通学生类）

* **继承**：ChengJiMember
* **成员变量**：
  + chengji：map<科目, 成绩>
* **实现基类所有虚函数**

### 2.3 PangTingStudent（旁听学生类）

* **继承**：ChengJiMember
* **成员变量**：
  + chengji：map<科目, 成绩>
* **实现基类所有虚函数**

### 2.4 ChengJiGuanLi（成绩管理类）

* **核心功能**：
  + 学生管理（添加、删除、查找）
  + 科目管理（添加、删除）
  + 成绩管理（修改）
  + 数据展示（支持多种排序）
* **数据结构**：
  + xueshengList：学生列表（unique\_ptr智能指针）
  + kemuList：科目列表
* **关键方法**：
  + addStudent()：添加学生（区分普通/旁听）
  + delStudent()：按学号删除学生
  + findStudent()：按学号或姓名查找学生
  + display()：展示成绩表单（支持按学号或科目成绩排序）
  + generateTestData()：生成测试数据

## 3. 功能设计

### 3.1 学生管理

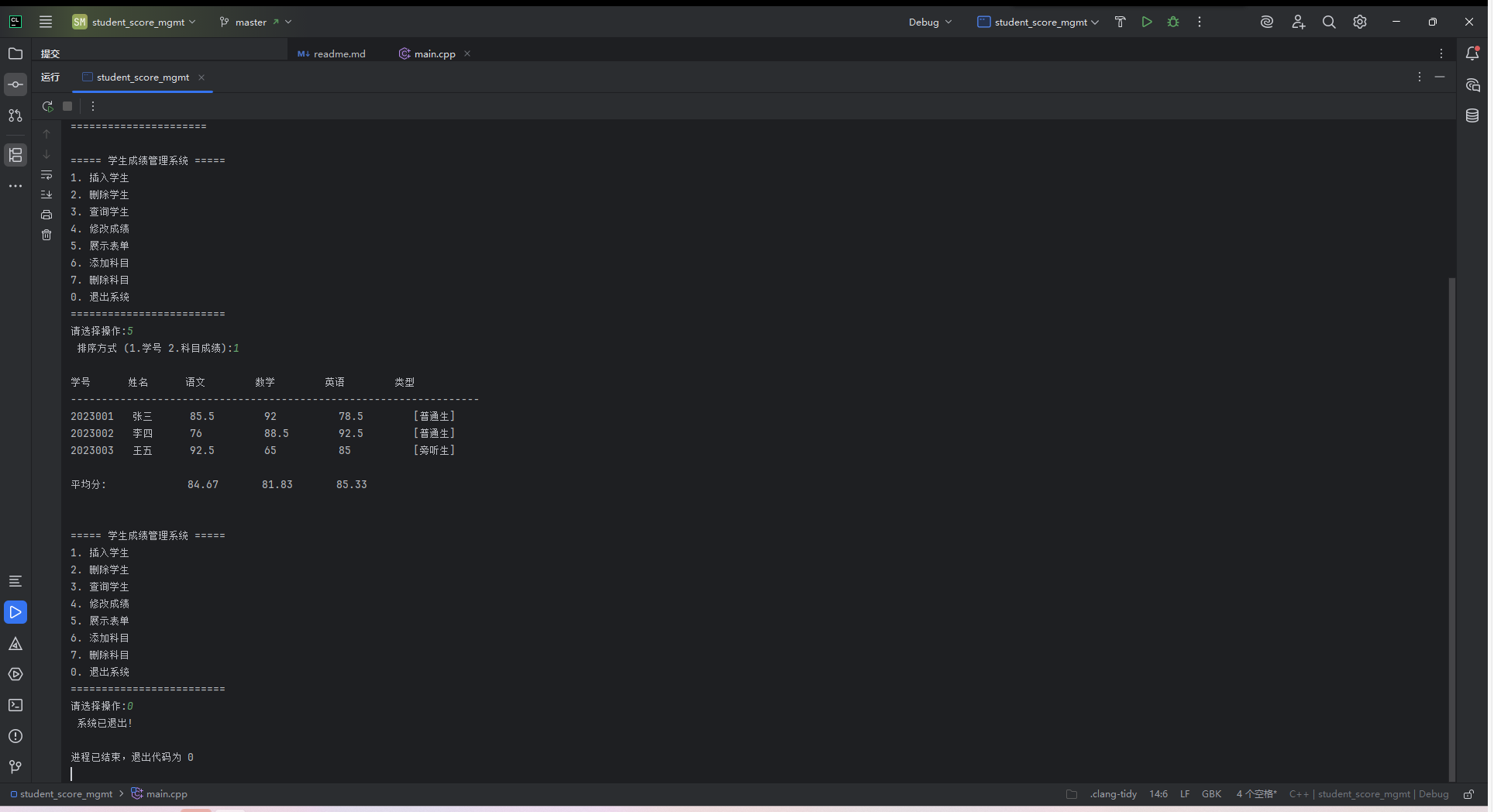
* 添加学生：需指定类型（普通/旁听）、姓名、学号
* 删除学生：按学号删除
* 查询学生：按学号或姓名查询
* 修改成绩：按学生标识+科目修改成绩

### 3.2 科目管理

* 添加科目：所有学生自动添加该科目
* 删除科目：所有学生自动删除该科目

### 3.3 数据展示

* 表格化显示：学号、姓名、各科成绩、学生类型
* 排序方式：
  + 默认按学号排序
  + 可选按指定科目成绩降序排序
* 统计功能：显示各科目平均分



## 4. 用户界面

* 控制台菜单驱动界面

===== 学生成绩管理系统 =====   
1. 插入学生   
2. 删除学生   
3. 查询学生   
4. 修改成绩   
5. 展示表单   
6. 添加科目   
7. 删除科目   
0. 退出系统   
=========================   
请选择操作:

## 5. 测试模式

* 通过#define TEST\_MODE 1开启，预置测试数据
* 3个科目：
  + 语文、数学、英语
* 3名学生：
  + 张三（普通生）：语文85.5，数学92，英语78.5
  + 李四（普通生）：语文76，数学88.5，英语92.5
  + 王五（旁听生）：语文92.5，数学65，英语85

## 6. 关键技术

1. **面向对象设计**：使用继承和多态处理不同类型学生
2. **智能指针**：使用unique\_ptr自动管理内存
3. **STL容器**：vector管理列表，map存储成绩
4. **灵活排序**：支持按学号和科目成绩排序
5. **格式化输出**：使用iomanip实现对齐表格

## 7. 测试用例

### 7.1.添加新学生

请选择操作:1   
 学生类型 (1.普通 2.旁听):1   
 姓名:赵六   
 学号:2025003   
 添加成功!

### 7.2.修改成绩

请选择操作:4   
 输入学号或姓名:2025003   
 输入科目:数学   
 输入新成绩:95   
 成绩修改成功!

### 7.3.按科目排序展示

请选择操作:5   
 排序方式 (1.学号 2.科目成绩):2   
 输入排序科目:数学   
   
学号 姓名 语文 数学 类型   
------------------------------------------------------   
2023001 张三 85.5 92 [普通生]   
2023002 李四 76 88.5 [普通生]   
2023003 王五 92.5 65 [旁听生]   
   
平均分: 84.67 81.83

### 7.4.添加新科目

请选择操作:6   
 输入新科目名称:物理   
 添加成功!

### 7.5.删除科目

请选择操作:7   
 输入删除科目名称:英语   
 删除成功!

### 7.6.删除学生

请选择操作:2   
 输入学号:2025003   
 删除成功!

### 7.7.查询学生

请选择操作:3   
 输入学号或姓名:张三   
 找到学生: 张三(2023001)

### 7.8.错误处理

请选择操作:4   
 输入学号或姓名:9999   
 输入科目:数学   
 输入新成绩:100   
 错误: 学生不存在!   
   
请选择操作:4   
 输入学号或姓名:2023001   
 输入科目:C++   
 输入新成绩:90   
 错误: 科目不存在!

### 7.9.退出系统

请选择操作:0   
 系统已退出!   
   
进程已结束，退出代码为 0

## 8.体会与总结

这次开发经历让我领悟到：**纸上蓝图终觉浅，代码落地始通幽**。调试时反复修正的焦灼，最终化为系统运行的流畅——这种从混沌到秩序的创造过程，比任何理论都更深刻地教会我软件工程的真谛：优雅背后是无数次迭代的坚持。当第一个成绩单成功打印的瞬间，我看到的不仅是数据，更是思维具象化的轨迹。

最珍贵的收获在于：**完美是迭代的副产品，而非起点**。每个报错都是系统与我的对话，每次修复都是认知的重构。这段旅程让我学会拥抱代码的不完美，在调试中培养与机器的默契，这正是开发者最真实的成长印记。