程序设计整体思路

1. **面向对象设计**：将成绩管理功能封装在 GradeManager 类中。
2. **数据结构选择**：使用 unordered_map 存储学号和成绩的对应关系。
3. **文件格式设计**：采用简单的文本文件格式，每行包含一个学号和对应成绩，用空格分隔，便于读写和人工查看。
4. **接口设计**：提供了清晰的接口，对应题目中的 8 个指令和相应功能。
5. **错误处理**：对可能的错误进行处理，如文件打开失败、学号不存在等，并提供用户友好的反馈。
6. **用户交互**：实现简洁明了的菜单界面，用户可以通过输入数字指令使用不同的功能。

测试案例设计思路

测试案例 1：基本操作测试

这个测试案例全面验证了系统的核心功能：

1. **添加功能**：添加三名学生（学号 1001, 1002, 1003）的成绩。
2. **显示功能**：验证添加后可以正确显示所有学生成绩。
3. **更新功能**：将学号 1002 的成绩从 87 更新为 92，验证更新功能。
4. **查询功能**：查询学号 1003 的成绩，验证查询功能正确。
5. **删除功能**：删除学号 1001，验证单个删除功能。
6. **文件保存功能**：将当前成绩表保存到文件，验证保存功能。
7. **清空功能**：清空所有记录，验证清空功能。
8. **文件加载功能**：从刚才保存的文件加载数据，验证文件读取功能。

该测试案例涵盖了系统的所有基本功能，确保每个功能都能正常工作，并且数据能够在内存和文件之间正确转换。

验证结果

测试案例执行结果验证了系统能够正确地：

1. 添加、更新、查询、删除和显示学生成绩
2. 清空成绩表单
3. 将成绩保存到文件并从文件中读取成绩

所有功能按预期工作，系统成功实现了所有需求。

程序运行实际结果

```
=== 测试案例1: 基本操作测试 ===
添加三名学生后的成绩表:
学号     成绩
1003     76
1001     95
1002     87
更新学号1002的成绩后:
学号     成绩
1003     76
1001     95
1002     92
学号1003的成绩为: 76
删除学号1001后的成绩表:
学号     成绩
1003     76
1002     92
保存成绩到test_grades.txt
清空后的成绩表:
成绩表单为空
从test_grades.txt加载成绩:
学号     成绩
1002     92
1003     76
=== 测试完成 ===


==== 学生成绩管理系统 ====
1: 从文件加载成绩
2: 添加学生成绩
3: 删除学生成绩
4: 清空所有成绩
5: 更新学生成绩
6: 查询学生成绩
7: 显示所有成绩
8: 保存成绩到文件
-1: 退出系统
请输入指令: -1
感谢使用，再见!
```