Programming-practice

C++程序设计作业

如果没有安装Markdown阅读器建议查看pdf版，体验更好。

第一部分

实践训练题目：

1. 输入一个长度小于100的字符串，判断其是否为回文串。
2. 输入长度为 n 的整形数组，分别使用选择和冒泡排序进行排序。
3. 输入长度为n的整形数组，使用快速排序或者归并排序进行排序。
4. 实现只支持加号、减号的计算器。
5. 实现二进制转10进制的转换器（输入二进制数，输出十进制数）
6. 输入一个日期（xxxx-xx-xx），计算n 天以后的日期。
7. 输入一个正整数，判断这个数是否为素数（质数）。
8. 输入长度小于10的字符串，输出所有子串。
9. 输入两个矩阵（分别为 m *n 及 n* q），编码输出两个矩阵相乘的结果。
10. 对一个长度为n的升序数列，用二分法进行查询。

完成情况：

选择了 `1` ， `2` ， `3` ， `5` ， `7` ， `10` 这六道题完成，此部分的程序位于 `Part1` 文件夹内。

程序的统一命名格式为 `practice_x.cpp` ，其中 `x` 为具体的题目编号，并同时将各程序的主要功能函数放置于 `practice_functions.cpp` 中，可以在 `main.cpp` 中进行测试和调用，通过更改 `CMakeLists.txt` 各题目程序也可独立运行。

第二部分

题目二选一

评分标准：用户界面不作为评分项（用Dos界面即可），评分项包括代码风格、注释完整性、功能完整性、文档完整性。

- 基于面向对象设计并实现学生成绩管理系统，用于统计单科成绩、加权分计算、年级排名等。设计发散功能可作为加分项。
- 自拟题目进行系统设计与实现，要求使用面向对象。

学生成绩管理系统

程序使用CLion开发，C++ 14.0标准，由于包含中文注释，部分文件使用 `GBK` 编码，使用 `UTF-8` 编码打开可能出现乱码，注意进行格式转换。

1.主体结构介绍

主要包含如下几个类：

1. GradeSystem
2. Method
3. Profile
4. StaticScore
5. Subject
6. GradeBean

下面将逐个进行大致介绍：

(1) `GradeSystem`

此类即是学生成绩管理系统类，用户和成绩系统进行的交互主要通过 `GradeSystem` 类之中的函数进行。

(2) `Method`

是一个抽象类，是成绩管理系统的基类，主要用于约束成绩管理系统所必须实现的一些功能，成绩管理系统的大多数函数是通过重写此类的函数进行的实现。

(3) `Profile`

类如其名，用于保存每个学生的档案，其中主要包含如下几个成员变量：

- `name`， `id` 分别用于保存学生的姓名和学号，
- `common`， `elcetive`，分别用于保存学生所选择的必修课和选修课信息，
- `num_com`，`num_elec`，分别用于保存学生所选择的必修课和选修课数量（此项并不是鸡肋），不直接采用类似于 `common.size()` 的方式获取是因为这两个成员变量都是 `private` 类型，无法被直接访问到，
- `overall_GPA`，`overall_grade`，`overall_credit` 用于保存此学生的平均总绩点，平均总加权和总学分数。

(4) `Staticscore`

用于实时统计各科目的平均成绩

(5) `Subject`

用于保存各个科目的信息，如学分、成绩、绩点、课程类型等

> `Profile` 类中的 `common` 和 `elective` 都是 `Subject` 类型的 `vector`

(6) `GradeBean`

此类的存在颇为尴尬，因为 `Profile` 和 `StaticScore` 以及 `Subject` 类的私有成员之间无法进行相互访问，所以在进行修改、删除成绩等操作时有很多不便，故使用此类用于传递信息

2.实现功能介绍

实现的功能主要包括（部分为发散功能）：

1. 学生成绩的输入（从键盘）
2. 学生成绩的输入（从文件）
3. 学生成绩查询（姓名，学号）
4. 单科及总成绩统计（包括平均绩点，加权分计算等）
5. 单科成绩排名及总成绩排名
6. 个人成绩统计
7. 个人成绩删除

8. 个人成绩修改
9. 成绩本地保存

部分实现细节介绍：

- 学生成绩键盘输入通过函数 `addInformationByTap()` 实现，可以进行单次多门成绩输入，只需注意登记结束时请以 “0 0 0 0” 格式结尾。
- 学生成绩文件输入通过函数 `addInformationByFile()` 实现，此处需要注意，文件的默认位置位于.exe所在的文件夹下，程序会提示.exe所处的绝对路径，只需在相应路径下依照格式新建输入文件即可。

  > 此处仍需注意：输入的.txt文件需要使用GBK编码，不然可能出现问题

- 学生成绩查询： `searchById()` 和 `searchByName()` 分别支持通过姓名和学号查询，在通过姓名查询时会输出所有重名学生的信息。
- 成绩统计：会展示该科目的平均加权、绩点等，同时会在下方展示所有有该科目成绩的学生的该科目成绩信息；如果选择查看总成绩则展示总成绩及各科目信息，还有学生成绩信息。
- 成绩排名：依照选定科目进行成绩排名。
- 个人成绩删除：通过函数 `deleteCourse()` 实现，可以一次删除多科成绩，循环读入，以0为结尾。
- 个人成绩修改：通过函数 `fixCourse()` 实现，可以修改个人成绩。
- 成绩本地保存：只要在退出程序时选择安全退出而非直接终止程序，程序便会在.exe文件所处目录下生成一份名为 `local_save.txt` 的文本文档，如原先存在，则会先删除原文档再重新生成，保存本次操作做出的修改。
- 程序每次启动后会先读取已经保存在本地的存档，这样就不会丢失之前进行的修改。
- 关于成绩统计：个人成绩和科目平均成绩的统计都是随着成绩的登记、修改和删除而实时改变的，所以一些函数的逻辑略显臃肿，有很多代码逻辑虽然近似，但因为使用DOS界面，为保证交互时的提示不同，所以并不能统一封装起来，略"脏"。
- 针对大部分可能出现的错误输入进行了优化和提示，增强了代码的健壮性，但仍可能存在一些没有发现的未知bug，但目前而言，假如不是故意卡数据，体验良好。
- 编译时可能会报许多Warning，因为部分实现并未完全按照C++ 14标准，但实际上很多逻辑错误并不会出现，在上一层函数中实际已经规避此错误，但编译器并不知道。
- 实现同一学生重复输入判断，如果一个学生此前已在系统内登记过某科成绩，重复输入时可实现覆盖并增加新科目。

3.使用提示

进入程序后会出现如下提示：

```
 std::cout <<
"*********************************************************************************" <<
std::endl;
    std::cout << "═══════════  ☆   学 生 成 绩 管 理 系 统     ☆  ═══════════
══" << std::endl;
    std::cout << "═══════★★★★★          ★★★★★★★          ★★★★★═══════
═" << std::endl;
    std::cout << "═══════════★  ☆          1.增加学生成绩          ☆  ★═══════
══" << std::endl;
    std::cout << "═══════════★  ☆          2.查询学生成绩          ☆  ★═══════
══" << std::endl;
    std::cout << "═══════════★  ☆          3.排序统计成绩          ☆  ★═══════
══" << std::endl;
    std::cout << "═══════════★  ☆          4.删除学生成绩          ☆  ★═══════
══" << std::endl;
    std::cout << "═══════════★  ☆          5.修改学生成绩          ☆  ★═══════
══" << std::endl;
    std::cout << "═══════════★  ☆          6.查看成绩总览          ☆  ★═══════
══" << std::endl;
    std::cout << "═══════════★  ☆          0.安全退出系统          ☆  ★═══════
══" << std::endl;
    std::cout << "Tip: 建议您使用安全退出，强行关闭程序可能会导致数据丢失。" << std::endl;
```

只需输入对应的编码即可进行相应操作。

功能及分类对应：

1. 学生成绩的输入（从键盘）————— 1
2. 学生成绩的输入（从文件）————— 1
3. 学生成绩查询（姓名，学号）————— 2
4. 单科及总成绩统计（包括平均绩点，加权分计算等）————— 3
5. 单科成绩排名及总成绩排名 ————— 3
6. 个人成绩统计 ————— 3
7. 个人成绩删除 ————— 4
8. 个人成绩修改 ————— 5
9. 成绩本地保存 ————— 0

> 查看成绩总览会按照学号排序展示所有学生的所有科目成绩， 也会输出所有当前已有科目的成绩统计。

4.已进行的优化

已经依照格式生成好了 `scores.txt` 和 `local_save.txt` ，保证在初次运行程序时即可体验所有操作，如不满足也可自行修改。

文件请一定依照格式，如果格式不对可能导致读入中止，程序会提供错误提示以便修改。

源代码存在一些必要注释。

已上传至GitHub

Git仓库地址： https://github.com/Septieme7/Programming-practice