课 程 设 计 报 告

课程名称 计算机程序设计基础 2

班级：无 12

学号：2021012644

姓名：李羿璇

2022 年 8 月 29 日

目录

1 系统需求分析

设计目的:

学生成绩管理系统记录了由教师所开的课程信息，同时记录了选课的学生在每一门课中所取得的成绩信息，并对两种信息都提供简单的增删查改功能，从而便于管理学生成绩。

功能分析:

1. 用户类应当包含用户的姓名、学号/工号等信息。用户应当能够自己选择注册具有本科生、研究生、教师等身份的账号。

2. 用户通过自己的学号/工号和密码登录系统，登录时应当自动判断账号类型，并且自动获取自己的账号类型所对应的权限。

3. 教师可以开课、可以停开选课人数较少的课，可以赋予学生成绩、可以修改学生成绩，可以查看全班成绩、排名。

4. 学生可以选课、可以将未被赋予成绩的课退课。同时，学生还可以查看自己的成绩，其中本科生查看成绩时只能查到绩点制成绩，研究生则只能查到百分制成绩。同时，本科生还可以查到自己的 GPA 以及 GPA 排名。

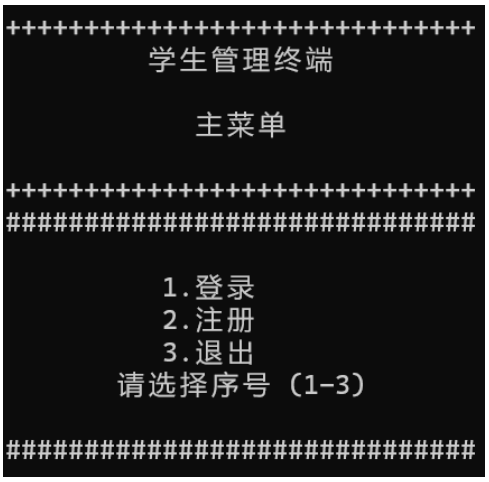
5. 此外，整个系统在退出时应当能正确地保存所有信息、启动时应当能正确地读取所有信息。

输入输出要求:

1. 由于系统较为繁琐复杂，输入过程中应当有清晰明确的引导，对于每一步的输入要求都应当有易懂的说明。由于每一步输入都只输入单独的信息，故无特别的输入格式。

2. 输出过程中，视需求情况不同，输出格式会有所变化。但总体而言，输出格式大概类似于图 1 中的两种情况

(a) 输出类型 1



(b) 输出类型 2

图 1: 输出格式示例

2 总体设计

学生成绩管理系统包含四个主要功能，分别为注册、登录、教师操作端和学生操作端。同时，教师操作端包含查看成绩、打分、开课、停开课四个功能；学生操作端则包含查看成绩、选课、退课三个功能。在各个操作功能中，用户都应受到明确清晰的提示，然后按照提示进行操作。具体的功能设计及模块图见图 2:

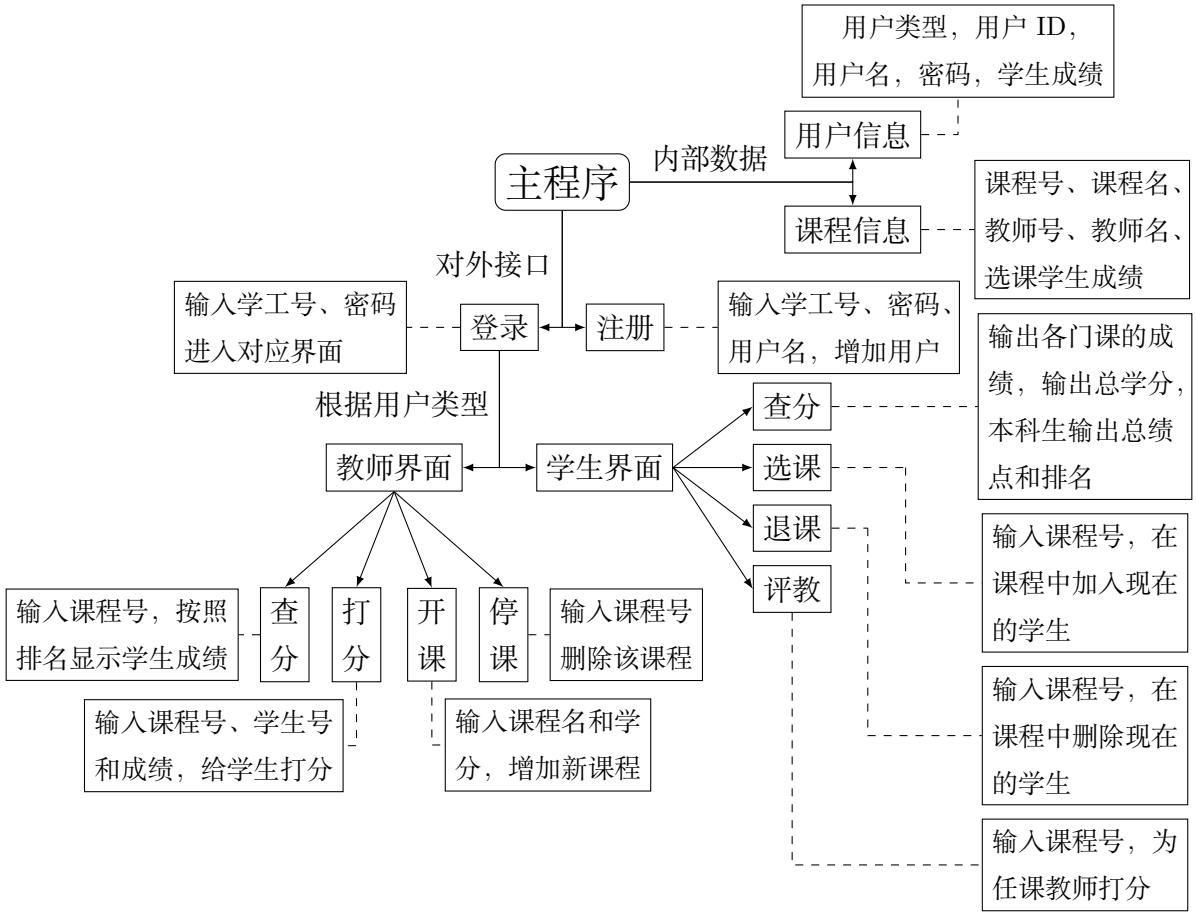


图 2: 系统功能设计及模块图

3 详细设计

数据文件:

具体程序设计中，较为简单的办法是采用 protobuf、boost 库等对数据进行序列化，然后将数据保存在 json 文件中。但是为了减少对外部库的依赖，最终采用二进制文件的方式进行数据文件的保存。

类层次图:

主进程放在 UI 类中，UI 类下包含 User 类和 Course 类作为数据成员。User 类派生出 Student 类和 Teacher 类，Student 类又派生出 UnderGraduate 类和 PostGraduate 类，Student 类下另有 GradePoint 类作为数据成员；Course 类下另有 Score 类作为数据成员。具体来讲，示意图见图 3。

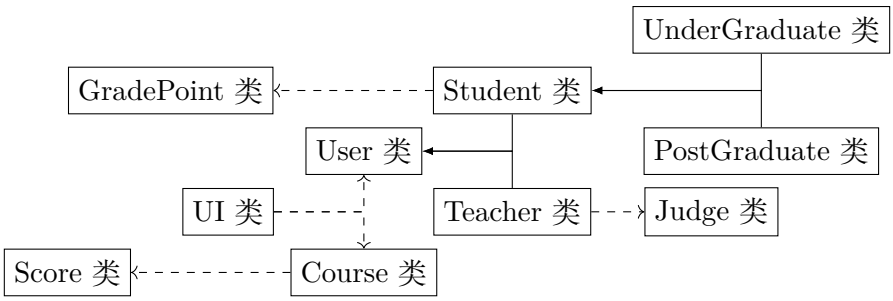


图 3: 类的层次图

界面设计和各功能模块实现:

界面设计如输入输出要求中的部分，采用 iostream 和 iomanip 进行输出。

UI 类是程序的主要进程，绝大部分功能模块都在 UI 类中实现，同时所有的交互输出都在 UI 类中完成、其他类中不包括输入输出；具体到对类内私有成员操作时，由 UI 类调用各类内部的公开成员函数来完成。

UML **类图**:

UML 类图见图 4、图 5。特别说明：为了简略起见，所有用于从外部获取类内数据、没有任何计算而只是单纯返回的函数均在 UML 图中省略。同时，抽象类中作为接口的纯虚函数，在某一派生类中只做简单继承、不做任何有实际意义重写的情况下，派生类中的该函数也在 UML 图中省略。

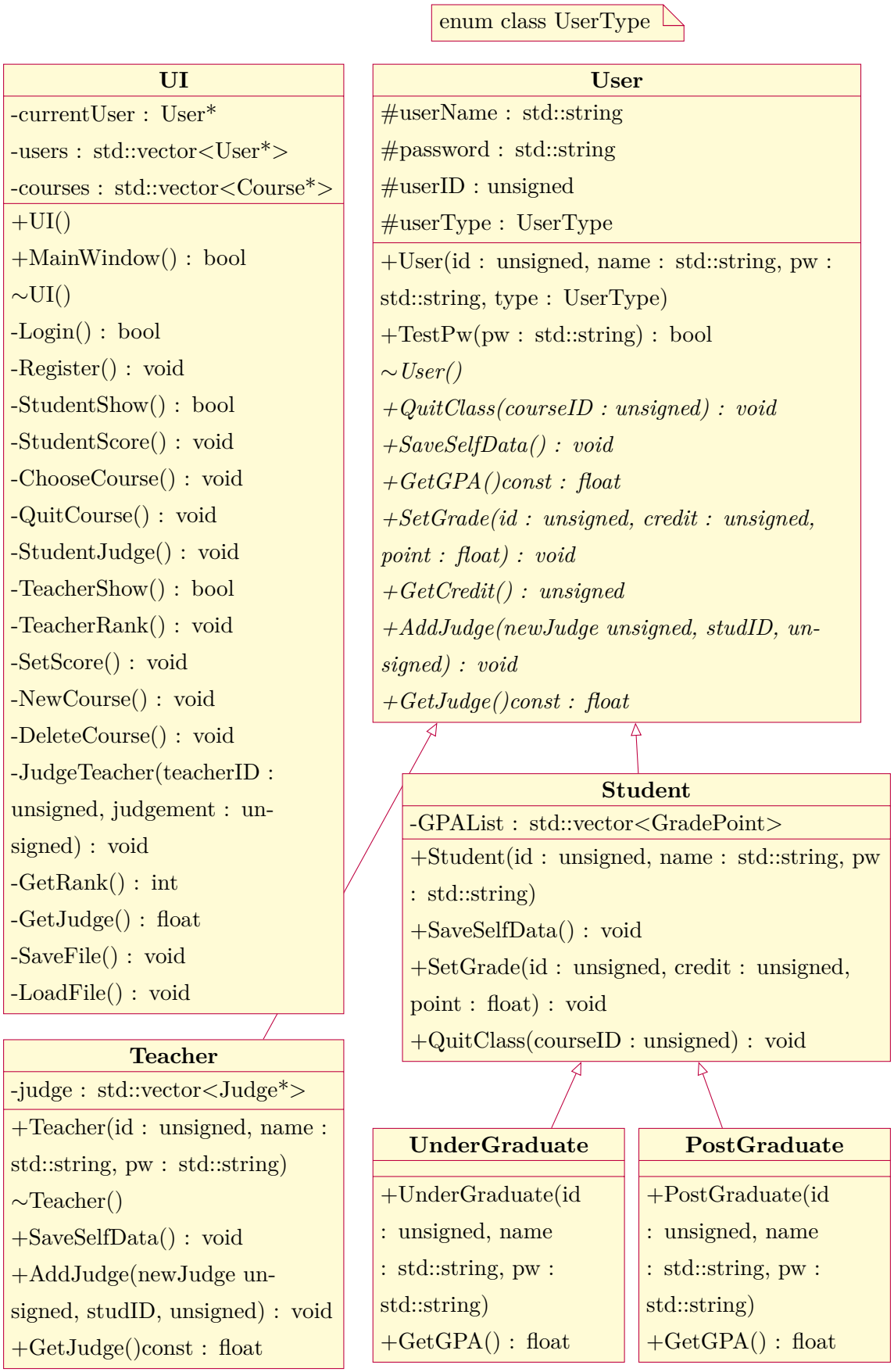

enum class UserType
UI
-currentUser : User*
-users : std::vector<User*>
-courses : std::vector<Course*>
+UI()
+MainWindow() : bool
~UI()
-Login() : bool
-Register() : void
-StudentShow() : bool
-StudentScore() : void
-ChooseCourse() : void
-QuitCourse() : void
-StudentJudge() : void
-TeacherShow() : bool
-TeacherRank() : void
-SetScore() : void
-NewCourse() : void
-DeleteCourse() : void
-JudgeTeacher(teacherID : unsigned, judgement : unsigned) : void
-GetRank() : int
-GetJudge() : float
-SaveFile() : void
-LoadFile() : void
User
#userName : std::string
#password : std::string
#userID : unsigned
#userType : UserType
+User(id : unsigned, name : std::string, pw : std::string, type : UserType)
+TestPw(pw : std::string) : bool
~User()
+QuitClass(courseID : unsigned) : void
+SaveSelfData() : void
+GetGPA()const : float
+SetGrade(id : unsigned, credit : unsigned, point : float) : void
+GetCredit() : unsigned
+AddJudge(newJudge unsigned, studID, unsigned) : void
+GetJudge()const : float
Student
-GPAList : std::vector<GradePoint>
+Student(id : unsigned, name : std::string, pw : std::string)
+SaveSelfData() : void
+SetGrade(id : unsigned, credit : unsigned, point : float) : void
+QuitClass(courseID : unsigned) : void
Teacher
-judge : std::vector<Judge*>
+Teacher(id : unsigned, name : std::string, pw : std::string)
~Teacher()
+SaveSelfData() : void
+AddJudge(newJudge unsigned, studID, unsigned) : void
+GetJudge()const : float
UnderGraduate
+UnderGraduate(id : unsigned, name : std::string, pw : std::string)
+GetGPA() : float
PostGraduate
+PostGraduate(id : unsigned, name : std::string, pw : std::string)
+GetGPA() : float

图 4: 类的 UML 图 1

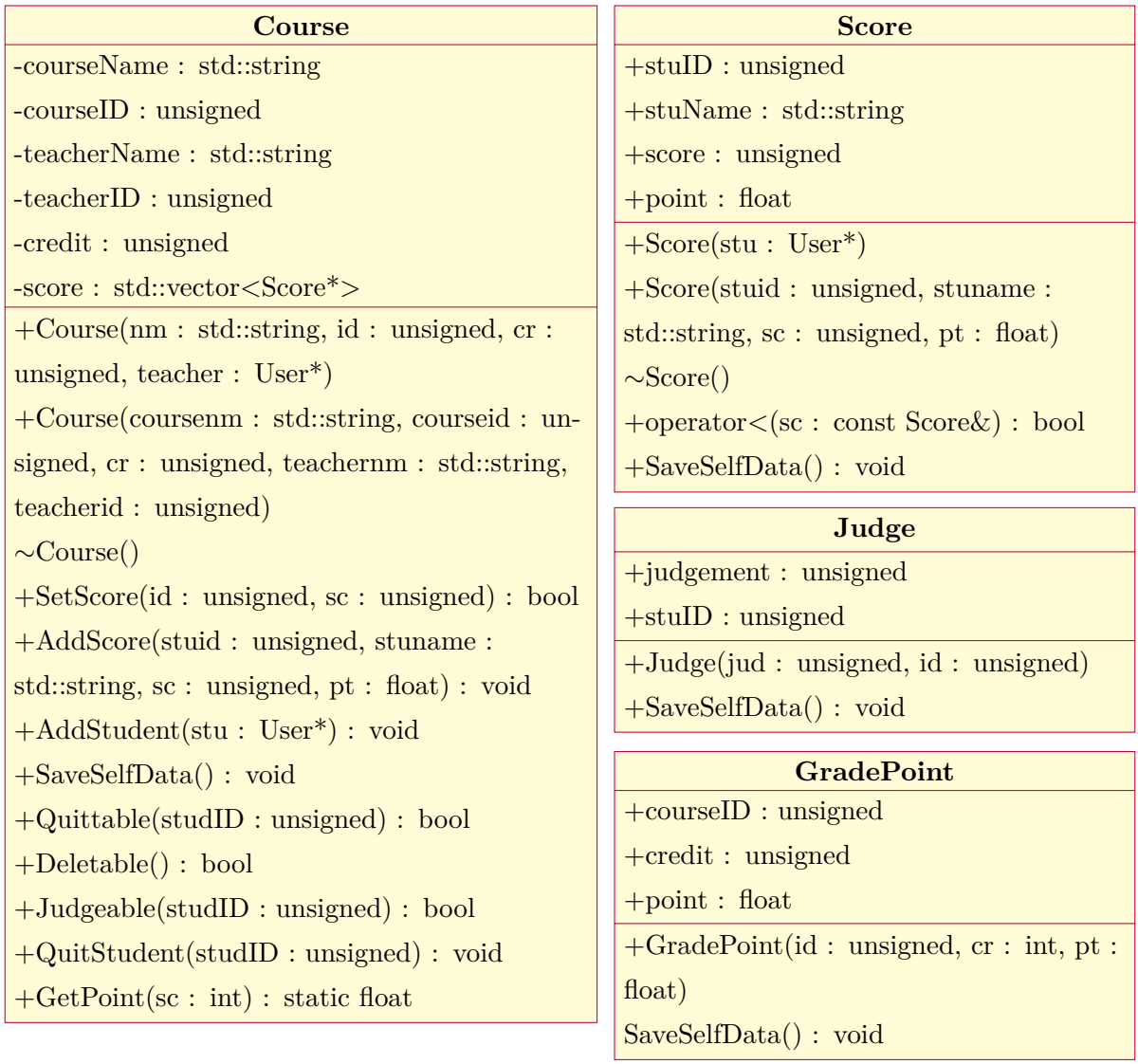


图 5: 类的 UML 图 2

4 系统调试

程序编写完成之后，我对整体程序进行了调试和运行，发现文件读写部分出现了问题。经检查，我发现自己是直接对整个 UI 类进行文件读写，而 UI 类中存放的数据成员则都是以指针形式存放的，对指针进行读写显然是没有意义的。于是我对程序进行了修改，改为遍历整个数组并依次读写指针指向的位置，但是依然存在一定的问题。经过思考，我注意到这是因为 users 数组中的所有成员均为 User*，但是这些基类指针指向了不同派生类的对象，这些派生类的大小是不同的。于是我在各个派生类中分别写了不同的文件读写函数，但是此时依然存在问题。经过一番查阅资料，我又发现 STL 的容器本质是一些指针，STL 容器如 std::string, std::vector 是不能直接进行读写的，而 User 类和 Course 类中又都包含 STL 容器的数据成员，故这些类都是不能直接读写的。为此，我大规模修改了文件读写部分的程序。在保存时，将各个类对象

内的信息手动序列化，并逐个逐条写入到二进制文件当中。读取时，则需要重新用 new 分配这些对象的内存，并逐一 push_back 到动态数组当中。经过这样的修改后，我的程序最终能够正常实现预想的功能。

总体来讲，在调试过程中，预先设计的程序逻辑是完善的。出现错误一方面是因为我自己过于想当然、没有仔细思考；另一方面也是因为我对 STL 的相关知识理解并不透彻。但由于我对程序的模块化设计相对较好，使我能够比较快速地定位并解决问题。

5 结果分析

由于采用二进制文件进行读写，故不能手动创造数据文件，数据文件会在程序进入时自动加载，退出时自动保存在 Data.dat。本次测试时所使用的数据及代码都可在下载地址处下载。

使用样例:

首先打开程序，显示界面如图 6(a)；选择注册，按照要求输入（如图 6(b)）之后我们就创建了一个新用户。这个用户是一名本科生。再选择登录，按照要求输入，如图 6(c)。

(a) 初始界面 (b) 注册 (c) 登录

图 6: 注册与登录

如此，我们就进入了学生操作界面（如图 7(a)）。选择选课，可以看到现在有两门课供我们选择，我们同时能看到开课教师和这位教师的教评分数（如图 7(b)）。我们选择一门课后，再进入退课界面，可以看到此时这门课已经出现在了我们的可退课列表里（尚未结课），而没有选择的另一门课则没有出现（如图 7(c)）。我们退掉这门课后，这门课会重新出现在可选课程列表里，将两门课都选上，随后选择退出，重新回到主界面。

(a) 学生界面 (b) 选课 (c) 退课

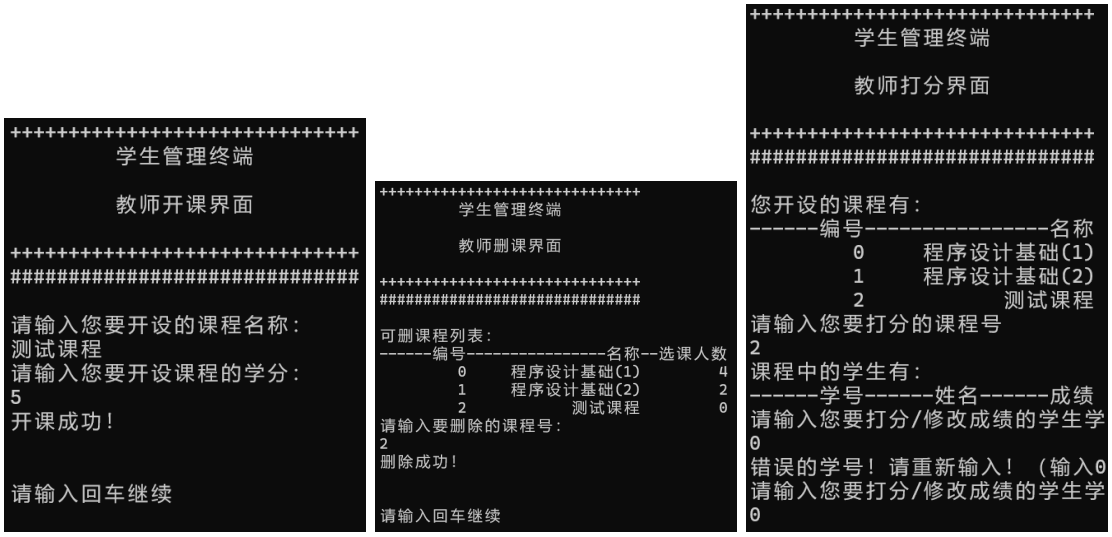
图 7: 选课与退课

输入事先注册好的教师账号密码后，我们成功进入了教师操作端（图 8(a)），选择查看成绩并输入课程号，可以查看当前所有选择该课程的学生的成绩排名（图 8(b)）。随后可以进入打分界面给测试用户打分（图 8(c)）。

(a) 教师界面 (b) 查分 (c) 打分

图 8: 教师查分与打分

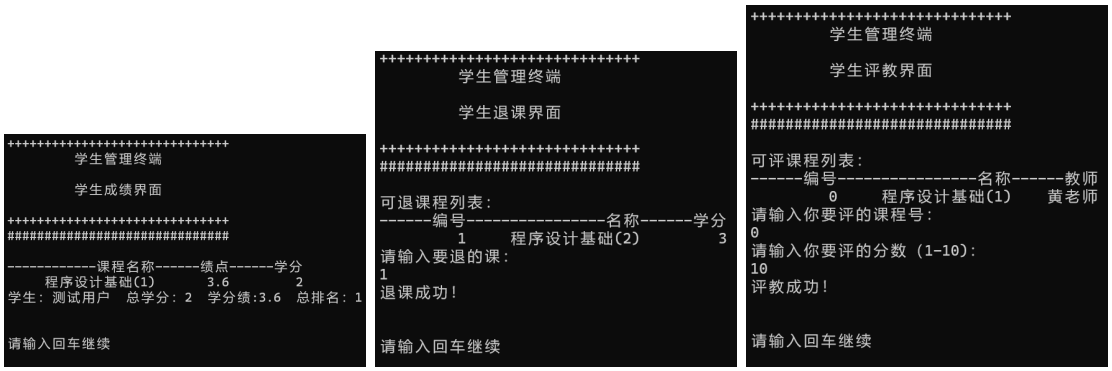
我们还可以选择新开一门课（图 9(a)），在打分界面看到这门课的选项（图 9(b)），虽然并没有可打的分，再将这门课删除掉（图 9(c)）。



(a) 开课 (b) 再打分 (c) 停开课

图 9: 教师开课与停开课

我们再回到刚才新建的测试用户，选择查看成绩，可以看到我们这门课的绩点、总修的学分和总的绩点（图 10(a)）。接下来选择退课，可以发现刚才被打好分的课已经不能退了，而另一门课还可以退（图 10(b)）。再回到。最后来到教评界面，可以给这位授课老师打分。要注意的是只能给一位老师打一个分，而不能每一门课都打一次分（图 10(c)）。



(a) 查分 (b) 退课 (c) 评教

图 10: 学生查分与评教

以上是本程序的主要功能介绍和使用说明。

6 总结

本次大作业的最终完成效果我还是比较满意的，但是最终提交的版本其实经历了一次重做。究其原因，是因为我一开始对虚函数的多态性没有更好的理解，总希望滥用，在输入输出的过

程中也用上虚函数的多态性。交互界面和数据处理没有很好地分开，从而写出了类似这样的不同类之间循环包含、耦合严重的代码：

```
class User
{//...
    UI* ui;
};
class UI
{//...
    std::vector<User*> users;
};
```

这样的代码显然是不合理的。思考之后，我决定将代码重构，争取将交互界面与数据处理、存储之间分离，User 类等只负责作为存储数据的工具，并不负责输出，于是就形成了现在的代码。其实现有代码同样不完美，它的前后端分离程度依然不够强。同时多态性只体现在了 User 类中而不体现在 UI 类中，这也是存在的一定问题。如果我们使用抽象类 UI 类派生出 StudentUI，TeacherUI，应当能解决这个问题。

此外，这个系统还有一些不完善的地方。首先，系统中没有做出超级管理员一类的权限，相关的接口有待补充。其次，输入之间的错误输入检测还不太完善，对错误输入只检测了一些常见的错误方式，如在输入 float 的场景输入 char 这种类型错误则并未完全检测，这也是日后应当完善的点。

总之，这次 C++ 大作业课程设计在很大程度上锻炼了我的编程能力，使我有了编写中型项目的经验。同时，这次大作业的第一次失误也让我充分认识到了写代码的过程中应当先有明确的计划再开始写，同时写的过程也要勤加思考，切忌心急、最后给自己徒增调试、DEBUG 的困难。

附录：源程序清单

UI.h

```
/*
* 文件名：UI.h
* 用途：定义 UI 类，是整个管理系统的主进程
* 作者：2021012644 李羿璇
*/

#pragma once
```

```
8  #include<iostream>
9  #include<vector>
10
11 class User;
12 class Course;
13
14 class UI
15 {
16 public:
17     UI(); //构造函数，读取数据
18     ~UI(); //析构调用，保存数据并释放内存
19     bool MainWindow(); //主窗口
20
21 private:
22     bool Login(); //提供登录功能
23     bool Register(); //提供注册功能
24     bool StudentShow(); //提供学生端的操作界面
25     void StudentScore(); //为学生提供查询成绩界面
26     void ChooseCourse(); //为学生提供选课界面
27     void QuitCourse(); //为学生提供退课界面
28     void StudentJudge(); //为学生提供评教界面
29     bool TeacherShow(); //提供教师端的操作界面
30     void TeacherRank(); //为教师提供查询成绩界面
31     void SetScore(); //为教师提供打分界面
32     void NewCourse(); //为教师提供开课界面
33     void DeleteCourse(); //为教师提供删课界面
34     float GetJudge(unsigned teacherID); //void ChooseCourse(); 的附属函数
35     int GetRank(); //void StudentScore(); 的附属函数，为学生提供 GPA 排名查询的
       ↪ 接口
36     void JudgeTeacher(unsigned, unsigned); //void StudentJudge(); 的附属函数
37     void SaveFile(); //在退出程序时调用，保存所有数据
38     void LoadFile(); //在启动程序时调用，加载所有数据
39
40     User* currentUser; //当前使用的用户，用于做权限判断
41     std::vector<User*> users; //用户信息储存数组
42     std::vector<Course*> courses; //课程信息储存数组
43
```

```
};
```

Course.h

```
/*
* 文件名：Course.h
* 用途：定义 Score 类、Course 类，提供课程信息的保存
* 作者：2021012644 李羿璇
*/

#pragma once
#include<iostream>
#include<vector>

class User;

struct Score
{
    Score(User*); //用于选课
    Score(unsigned, std::string, unsigned, float); //用于文件读取
    ~Score();
    bool operator<(const Score& sc)
    {
        return score < sc.score;
    }
    void SaveSelfData();
    unsigned stuID;
    std::string stuName;
    unsigned score;
    float point;
};

class Course
{
public:
    Course() = delete;
    ~Course();
```

```
34      Course(std::string, unsigned, unsigned, User*);
35      Course(std::string, unsigned, unsigned, std::string, unsigned);
36      unsigned GetID()const { return courseID; }
37      std::string GetTeacherName()const { return teacherName; }
38      unsigned GetTeacherID()const { return teacherID; }
39      unsigned GetCredit()const { return credit; }
40      unsigned GetNum()const { return score.size(); }
41      std::string GetName()const { return courseName; }
42      std::vector<Score*> GetScore()const { return score; }
43      bool SetScore(unsigned, unsigned); //设置对应学号的分数
44      void AddScore(unsigned, std::string, unsigned, float); //用于文件读写，直接
        ↪ 增加一个 score
45      void AddStudent(User*); //用于选课
46      void SaveSelfData(); //用于保存自身数据
47      bool Quittable(unsigned); //用于判断某学生的课是否可退
48      bool Deletable(); //用于判断课是否可删
49      bool Judgeable(unsigned); //用于判断课是否可评
50      void QuitStudent(unsigned); //用于退课
51      static float GetPoint(int); //用于百分制转绩点
52
53  private:
54      std::string courseName; //保存课程名
55      unsigned courseID; //保存课程号
56      std::string teacherName; //保存教师名（便于输出）
57      unsigned teacherID; //保存教师号
58      unsigned credit; //学分
59      std::vector<Score*> score; //学生成绩
60
61  };
```

User.h

```
1   /*
2   * 文件名：User.h
3   * 用途：定义枚举类 UserType 类；定义抽象类 User 类，派生出 Teacher 类和 Student
    ↪ 类，Student 类派生出 UnderGraduate 类和 PostGraduate 类；定义 GradePoint 类，
    ↪ 定义 Judge 类。用于保存用户信息
```

```
4  * 作者: 2021012644 李羿璇
5  */
6
7  #pragma once
8  #include<vector>
9  #include<iostream>
10 #include<fstream>
11 class Course;
12
13 enum class UserType //用户类型
14 {
15     Teacher,
16     UnderGraduate,
17     PostGraduate,
18 };
19
20 class User //User 类作为基类，提供各类接口供派生类实现
21 {
22 public:
23     User() = delete;
24     User(unsigned id, std::string name, std::string pw, UserType type) {
       ↪  password = pw; userName = name; userID = id; userType = type; }
25     virtual ~User();
26     unsigned GetID()const { return userID; } //返回 ID
27     UserType GetType()const { return userType; } //返回自身类型
28     std::string GetName()const { return userName; } //返回名称
29     bool TestPw(std::string pw)const { return pw == password; } //判断密码是否
       ↪  正确
30     virtual void QuitClass(unsigned courseID) = 0; //退课
31     virtual void SaveSelfData() = 0; //用于保存自身的数据，由于类之间继承复杂以
       ↪  及 stl 不允许直接读写，故采用手动序列化的方式读写
32     virtual float GetGPA()const = 0; //计算自己的 GPA
33     virtual void SetGrade(unsigned id, unsigned credit, float point) = 0; //设
       ↪  置课程号为 id 的课程的学分/分数
34     virtual unsigned GetCredit() = 0; //计算已修课程的 GPA
35     virtual void AddJudge(unsigned, unsigned) = 0;
36     virtual float GetJudge()const = 0;
```

```
37
38  protected:
39      std::string userName;
40      std::string password;
41      unsigned userID;
42      UserType userType;
43
44  };
45
46  struct Judge
47  {
48      Judge(unsigned jud, unsigned id) { judgement = jud, stuID = id; }
49      void SaveSelfData();
50      unsigned judgement;
51      unsigned stuID; //为了防止重复评分
52  };
53
54  class Teacher :virtual public User
55  {
56  public:
57      Teacher() = delete;
58      Teacher(unsigned id, std::string name, std::string pw) :User(id, name, pw,
        ↪  UserType::Teacher) {}
59      ~Teacher();
60      void SaveSelfData()override;
61      void SetGrade(unsigned id, unsigned credit, float point)override {}
62      void QuitClass(unsigned)override {}
63      float GetGPA()const override{ return -1; }
64      unsigned GetCredit()override { return 0; }
65      void AddJudge(unsigned newJudge, unsigned studID)override;
66      float GetJudge()const override;
67
68  private:
69      std::vector<Judge*> judge;
70  };
71
72  struct GradePoint
```

```
73  {
74      GradePoint(unsigned id, int cr, float pt) { courseID = id, credit = cr,
        ↪  point = pt; }
75      void SaveSelfData();
76      unsigned courseID;
77      unsigned credit;
78      float point;
79  };
80
81  class Student :virtual public User
82  {
83  public:
84      Student() = delete;
85      Student(unsigned id, std::string name, std::string pw, UserType type)
        ↪  :User(id, name, pw, type) {}
86      ~Student();
87      void SaveSelfData()override;
88      void SetGrade(unsigned id, unsigned credit, float point)override;
89      void QuitClass(unsigned)override;
90      unsigned GetCredit()override;
91      void AddJudge(unsigned, unsigned)override {}
92      float GetJudge()const override { return 0; }
93
94  protected:
95      std::vector<GradePoint*> GPAList; //为了方便计算 GPA 而无须大规模遍历
96  };
97
98  class UnderGraduate :public Student
99  {
100 public:
101     UnderGraduate() = delete;
102     UnderGraduate(unsigned id, std::string name, std::string pw) :User(id,
        ↪  name, pw, UserType::UnderGraduate), Student(id, name, pw,
        ↪  UserType::UnderGraduate) {}
103     float GetGPA()const override;
104
105 };
```

```
106
107 class PostGraduate :virtual public Student
108 {
109 public:
110     PostGraduate() = delete;
111     PostGraduate(unsigned id, std::string name, std::string pw) :User(id,
        ↪ name, pw, UserType::PostGraduate), Student(id, name, pw,
        ↪ UserType::PostGraduate) {}
112     float GetGPA()const override { return -1; }
113
114 };
```

UI.cpp

```
1  /*
2  * 文件名: UI.cpp
3  * 用途: 完成 UI 类中的函数定义, 兼记录最后的 courseID 信息
4  * 作者: 2021012644 李羿璇
5  */
6
7  #include<iostream>
8  #include<iomanip>
9  #include<algorithm>
10 #include<fstream>
11 #include<chrono>
12 #include<thread>
13 #include"UI.h"
14 #include"User.h"
15 #include"Course.h"
16 using namespace std;
17
18 unsigned courseID = 0;
19
20 UI::UI()
21 {
22     LoadFile();
23 }
```

```
24
25  UI::~UI()
26  {
27      SaveFile();
28      currentUser = nullptr;
29      for (auto itr = users.begin(); itr != users.end(); itr++)
30      {
31          if ((*itr) != nullptr)
32          {
33              delete *itr;
34              *itr = nullptr;
35          }
36      }
37      for (auto itr = courses.begin(); itr != courses.end(); itr++)
38      {
39          if ((*itr) != nullptr)
40          {
41              delete* itr;
42              *itr = nullptr;
43          }
44      }
45      users.clear();
46      users.shrink_to_fit();
47      courses.clear();
48      courses.shrink_to_fit();
49  }
50
51  bool UI::MainWindow()
52  {
53      //for (int i = 0; i < 10; i++)
54      //^^Icout << endl;
55      cout << setw(30) << setfill('+') << "+" << endl;
56      cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
57      cout << setw(18) << setfill(' ') << " 主菜单" << endl << endl;
58      cout << setw(30) << setfill('+') << "+" << endl;
59      cout << setw(30) << setfill('#') << "#" << setfill(' ')<< endl << endl;
60      cout << setw(16) << "1. 登录" << endl;
```

```
61      cout << setw(16) << "2. 注册" << endl;
62      cout << setw(16) << "3. 退出" << endl;
63      cout << setw(23) << " 请选择序号 (1-3)" << endl << endl;
64      cout << setw(30) << setfill('#') << "#" << endl;
65      int ChosenNumber;
66      cin >> ChosenNumber;
67      while (ChosenNumber > 3 || ChosenNumber < 1)
68      {
69          cin.clear();
70          cin.ignore();
71          cout << " 输入错误! 请重新输入! " << endl;
72          cin >> ChosenNumber;
73      }
74      for (int i = 0; i < 10; i++)
75          cout << endl;
76      switch (ChosenNumber)
77      {
78      case 1:
79          if (Login())
80          {
81              switch (currentUser->GetType())
82              {
83              case UserType::Teacher:
84                  while (TeacherShow() == true);
85                  break;
86              case UserType::PostGraduate:
87              case UserType::UnderGraduate:
88                  while (StudentShow() == true);
89                  break;
90              }
91          }
92          return true;
93      case 2:
94          Register();
95          return true;
96      case 3:
97          return false;
```

```
98      }
99  }
100
101 bool UI::Login()
102 {
103     unsigned id;
104     cout << setw(30) << setfill('+') << "+" << endl;
105     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
106     cout << setw(19) << setfill(' ') << " 登录界面" << endl << endl;
107     cout << setw(30) << setfill('+') << "+" << endl;
108     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
109
110     cout << " 请输入您的学工号: " << endl;
111     cin >> id;
112     for (auto item : users)
113         if (item->GetID() == id)
114         {
115             string pw;
116             while (true)
117             {
118                 cout << " 请输入您的密码: " << endl;
119                 cin >> pw;
120                 if (item->TestPw(pw) == true)
121                 {
122                     currentUser = item;
123                     cout << endl << endl << " 欢迎回来, " <<
                        ↪  currentUser->GetName() << endl << endl;
124                     this_thread::sleep_for(chrono::seconds(1));
125
126                     return true;
127                 }
128                 else
129                 {
130                     cout << " 您输入了错误的密码, 请重新输入! 按 1 退出, 按其他任
                        ↪  意键重试" << endl;
131                     char r;
132                     cin >> r;
```

```
133                     if (r == 1)return false;
134                     else;
135                 }
136             }
137         }
138     cout << " 没有找到您输入的学工号! 请先注册" << endl;
139     return false;
140 }
141 
142 bool UI::Register()
143 {
144     unsigned id;
145     int type = 0;
146     cout << setw(30) << setfill('+') << "+" << endl;
147     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
148     cout << setw(19) << setfill(' ') << " 注册界面" << endl << endl;
149     cout << setw(30) << setfill('+') << "+" << endl;
150     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
151 
152     cout << " 请输入你的身份: " << endl;
153     cout << "1. 老师" << endl;
154     cout << "2. 本科生" << endl;
155     cout << "3. 研究生" << endl;
156     while (type != 1 && type != 2 && type != 3)
157     {
158         cin >> type;
159         if (type == 1 || type == 2 || type == 3)break;
160         else cout << " 输入错误! 请重新输入! " << endl;
161     }
162     while (true)
163     {
164         bool flag = false;
165         cout << " 请输入你的学工号: " << endl;
166         cin >> id;
167         if (id < 2000000000 || id > 3000000000)
168         {
169             flag = true;
```

```
170             char r;
171             cout << " 您输入了错误的学工号，请重新输入！按 1 退出，按其他任意键重
                ↪  试" << endl;
172             cin >> r;
173             if (r == '1')return false;
174             else continue;
175         }
176         else
177         {
178             for (auto item : users)
179             {
180                 if (item->GetID() == id)
181                 {
182                     flag = true;
183                     char r;
184                     cout << " 这个账号已经注册过了，请重新输入！按 1 退出，按其他
                        ↪  任意键重试。" << endl;
185                     cin >> r;
186                     if (r == '1')return false;
187                     else;
188                 }
189             }
190         }
191         if (flag == false) break;
192     }
193     std::string pw = "1";
194     while (pw.length() < 8)
195     {
196         cout << " 请输入一个至少为 8 位的密码： " << endl;
197         cin >> pw;
198     }
199     std::string nm;
200     while (nm.empty() == true)
201     {
202         cout << " 请输入你的姓名： " << endl;
203         cin >> nm;
204     }
```

```
205     User* newUser;
206     switch (type)
207     {
208     case 1:
209         newUser = new Teacher(id, nm, pw);
210         break;
211     case 2:
212         newUser = new UnderGraduate(id, nm, pw);
213         break;
214     case 3:
215         newUser = new PostGraduate(id, nm, pw);
216         break;
217     }
218     users.push_back(newUser);
219     return true;
220 }
221
222 bool UI::StudentShow()
223 {
224     //for (int i = 0; i < 10; i++)
225     //^^Icout << endl;
226     cout << setw(30) << setfill('+') << "+" << endl;
227     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
228     cout << setw(18) << setfill(' ') << " 学生端" << endl << endl;
229     cout << setw(30) << setfill('+') << "+" << endl;
230     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
231
232     cout << setw(20) << "1. 查看成绩" << endl;
233     cout << setw(16) << "2. 选课" << endl;
234     cout << setw(16) << "3. 退课" << endl;
235     cout << setw(16) << "4. 评教" << endl;
236     cout << setw(16) << "5. 退出" << endl;
237     cout << setw(23) << " 请选择序号 (1-4)" << endl << endl;
238     cout << setw(30) << setfill('#') << "#" << endl;
239     int ChosenNumber;
240     cin >> ChosenNumber;
241     while (ChosenNumber > 5 || ChosenNumber < 1)
```

```
242     {
243         cout << " 输入错误! 请重新输入! " << endl;
244         cin >> ChosenNumber;
245     }
246     for (int i = 0; i < 10; i++)
247         cout << endl;
248     switch (ChosenNumber)
249     {
250     case 1:
251         StudentScore();
252         return true;
253     case 2:
254         ChooseCourse();
255         return true;
256     case 3:
257         QuitCourse();
258         return true;
259     case 4:
260         StudentJudge();
261         return true;
262     case 5:
263         currentUser = nullptr;
264         return false;
265     default:
266         return false;
267     }
268 }
269
270 void UI::StudentScore()
271 {
272     cout << setw(30) << setfill('+') << "+" << endl;
273     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
274     cout << setw(21) << setfill(' ') << " 学生成绩界面" << endl << endl;
275     cout << setw(30) << setfill('+') << "+" << endl;
276     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
277
278     if (currentUser->GetType() == UserType::UnderGraduate)
```

```
279     {
280         cout << setfill('-') << setw(20) << " 课程名称" << setw(10) << " 绩点"
            ↪  << setw(10) << " 学分" << endl;
281         for (auto item : courses)
282         {
283             for (auto score : item->GetScore())
284             {
285                 if (score->stuID == currentUser->GetID() && score->point !=
                    ↪  -1)
286                 {
287                     cout << setfill(' ') << setw(20) << item->GetName() <<
                        ↪  setw(10) << score->point << setw(10) <<
                        ↪  item->GetCredit() << endl;
288                     break;
289                 }
290             }
291         }
292         cout << setfill(' ') << " 学生: " << currentUser->GetName() << "  总学
            ↪  分: "<<currentUser->GetCredit()<< "  学分绩:" <<
            ↪  currentUser->GetGPA() << "  总排名: " << GetRank() << endl;
293     }
294     else if (currentUser->GetType() == UserType::PostGraduate)
295     {
296         cout << setfill('-') << setw(20) << " 课程名称" << setw(10) << " 成绩"
            ↪  << setw(10) << " 学分" << endl;
297         for (auto item : courses)
298         {
299             for (auto score : item->GetScore())
300             {
301                 if (score->stuID == currentUser->GetID())
302                 {
303                     cout << setfill(' ') << setw(20) << item->GetName() <<
                        ↪  setw(10) << score->score << setw(10) <<
                        ↪  item->GetCredit() << endl;
304                     break;
305                 }
306             }
```

```
307         }
308         cout << setfill(' ') << " 学生: " << currentUser->GetName() << "  总学
            ↪  分: " << currentUser->GetCredit() << endl;
309     }
310     cout << endl << endl << " 请输入回车继续" << endl;
311     cin.clear();
312     cin.ignore();
313     while (cin.get() != '\n');
314 }
315
316 void UI::ChooseCourse()
317 {
318     cout << setw(30) << setfill('+') << "+" << endl;
319     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
320     cout << setw(21) << setfill(' ') << " 学生选课界面" << endl << endl;
321     cout << setw(30) << setfill('+') << "+" << endl;
322     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
323
324     vector<Course*> selectableCourses;
325     cout << " 可选课程列表: " << endl;
326     cout << setfill('-') << setw(10) << " 编号" << setw(20) << " 名称" <<
        ↪  setw(10) << " 授课教师" << setw(5) << setprecision(4) << " 教评" <<
        ↪  setw(5) << " 学分" << endl;
327     for (auto item : courses)
328     {
329         bool flag = true;
330         for (auto& stu : item->GetScore())
331             if (stu->stuID == currentUser->GetID())
332             {
333                 flag = false;
334                 break;
335             }
336         if (flag == true)
337         {
```

```
338             cout << setfill(' ') << setw(10) << item->GetID() << setw(20) <<
                ↪  item->GetName() << setw(10) << item->GetTeacherName() <<
                ↪  setw(5) << GetJudge(item->GetTeacherID()) << setw(5) <<
                ↪  item->GetCredit() << endl;
339             selectableCourses.push_back(item);
340         }
341     }
342     cout << " 请输入你想选的课的序号（若都不想选，则随意输入一个错误的序号即可）: "
        ↪  << endl;
343     int choice;
344     cin >> choice;
345     for (auto item : selectableCourses)
346         if (choice == item->GetID())
347         {
348             item->AddStudent(currentUser);
349             currentUser->SetGrade(item->GetID(), item->GetCredit(), -1);
350             cout << " 选课成功！祝你取得好成绩！" << endl;
351             cout << endl << endl << " 请输入回车继续" << endl;
352             cin.clear();
353             cin.ignore();
354             while (cin.get() != '\n');
355             return;
356         }
357     cout << " 没有该课程，请重新选择课程！" << endl;
358     cout << endl << endl << " 请输入回车继续" << endl;
359     cin.clear();
360     cin.ignore();
361     while (cin.get() != '\n');
362     return;
363 }
364
365 void UI::QuitCourse()
366 {
367     cout << setw(30) << setfill('+') << "+" << endl;
368     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
369     cout << setw(21) << setfill(' ') << " 学生退课界面" << endl << endl;
370     cout << setw(30) << setfill('+') << "+" << endl;
```

```
371     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
372     cout << " 可退课程列表: " << endl;
373     cout << setfill('-') << setw(10) << " 编号" << setw(20) << " 名称" <<
        ↪  setw(10) << " 学分" << endl;
374     for (auto& item : courses)
375     {
376         if (item->Quittable(currentUser->GetID()) == true)
377         {
378             cout << setfill(' ') << setw(10) << item->GetID() << setw(20) <<
                ↪  item->GetName() << setw(10) << item->GetCredit() << endl;
379         }
380     }
381     cout << " 请输入要退的课: " << endl;
382     unsigned idToQuit;
383     cin >> idToQuit;
384     for (auto& item : courses)
385     {
386         if (item->Quittable(currentUser->GetID()) == true && item->GetID() ==
            ↪  idToQuit)
387         {
388             item->QuitStudent(currentUser->GetID());
389             currentUser->QuitClass(idToQuit);
390             cout << " 退课成功! " << endl;
391             cout << endl << endl << " 请输入回车继续" << endl;
392             cin.clear();
393             cin.ignore();
394             while (cin.get() != '\n');
395             return;
396         }
397     }
398     cout << " 错误的课程号! " << endl;
399     cout << endl << endl << " 请输入回车继续" << endl;
400     cin.clear();
401     cin.ignore();
402     while (cin.get() != '\n');
403     return;
404 }
```

```
405
406 void UI::StudentJudge()
407 {
408     cout << setw(30) << setfill('+') << "+" << endl;
409     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
410     cout << setw(21) << setfill(' ') << " 学生评教界面" << endl << endl;
411     cout << setw(30) << setfill('+') << "+" << endl;
412     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
413     cout << " 可评课程列表: " << endl;
414     cout << setfill('-') << setw(10) << " 编号" << setw(20) << " 名称" <<
        ↪  setw(10) << " 教师" << endl;
415     for (auto item : courses)
416     {
417         if(item->Judgeable(currentUser->GetID()) == true)
418             cout << setfill(' ') << setw(10) << item->GetID() << setw(20) <<
                ↪  item->GetName() << setw(10) << item->GetTeacherName() << endl;
419     }
420     cout << " 请输入你要评的课程号: " << endl;
421     unsigned idToJudge;
422     cin >> idToJudge;
423     for (auto item : courses)
424     {
425         if (item->GetID() == idToJudge &&
            ↪  item->Judgeable(currentUser->GetID()) == true)
426         {
427             cout << " 请输入你要评的分数 (1-10): " << endl;
428             unsigned scoreToJudge = 0;
429             cin >> scoreToJudge;
430             while (scoreToJudge > 10 || scoreToJudge < 1)
431             {
432                 cout << " 错误的分数! 请重新输入! " << endl << " 请输入您要给的分
                    ↪  数 (1-10): ";
433                 cin >> scoreToJudge;
434             }
435             JudgeTeacher(item->GetTeacherID(), scoreToJudge);
436             cout << " 评教成功! " << endl;
437             cout << endl << endl << " 请输入回车继续" << endl;
```

```
438             cin.clear();
439             cin.ignore();
440             while (cin.get() != '\n');
441             return;
442         }
443     }
444     cout << " 错误的课程号! " << endl;
445     cout << endl << endl << " 请输入回车继续" << endl;
446     cin.clear();
447     cin.ignore();
448     while (cin.get() != '\n');
449     return;
450 }
451
452 void UI::DeleteCourse()
453 {
454     cout << setw(30) << setfill('+') << "+" << endl;
455     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
456     cout << setw(21) << setfill(' ') << " 教师删课界面" << endl << endl;
457     cout << setw(30) << setfill('+') << "+" << endl;
458     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
459     cout << " 可删课程列表: " << endl;
460     cout << setfill('-') << setw(10) << " 编号" << setw(20) << " 名称" <<
    ↪  setw(10) << " 选课人数" << endl;
461     for (auto item : courses)
462     {
463         if(item->GetTeacherID() == currentUser->GetID() && item->Deletable()
        ↪  == true)
464             cout << setfill(' ') << setw(10) << item->GetID() << setw(20) <<
            ↪  item->GetName() << setw(10) << item->GetNum() << endl;
465     }
466     cout << " 请输入要删除的课程号: " << endl;
467     unsigned idToDel;
468     cin >> idToDel;
469     for (auto itr = courses.begin(); itr != courses.end(); itr++)
470     {
```

```
471             if ((*itr)->GetTeacherID() == currentUser->GetID() &&
                ↪  (*itr)->Deletable() == true && (*itr)->GetID() == idToDel)
472             {
473                 itr = courses.erase(itr);
474                 for (auto item : users)
475                     item->QuitClass(idToDel);
476                 cout << " 删除成功! " << endl;
477                 cout << endl << endl << " 请输入回车继续" << endl;
478                 cin.clear();
479                 cin.ignore();
480                 while (cin.get() != '\n');
481                 return;
482             }
483         }
484         cout << " 错误的课程号! " << endl;
485         cout << endl << endl << " 请输入回车继续" << endl;
486         cin.clear();
487         cin.ignore();
488         while (cin.get() != '\n');
489         return;
490     }
491
492     bool UI::TeacherShow()
493     {
494         //for (int i = 0; i < 10; i++)
495         //^^Icout << endl;
496         cout << setw(30) << setfill('+') << "+" << endl;
497         cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
498         cout << setw(18) << setfill(' ') << " 教师端" << endl << endl;
499         cout << setw(30) << setfill('+') << "+" << endl;
500         cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
501
502         cout << setw(20) << "1. 查看成绩" << endl;
503         cout << setw(16) << "2. 打分" << endl;
504         cout << setw(16) << "3. 开课" << endl;
505         cout << setw(16) << "4. 删课" << endl;
506         cout << setw(16) << "5. 退出" << endl;
```

```
507     cout << setw(23) << " 请选择序号 (1-5)" << endl << endl;
508     cout << setw(30) << setfill('#') << "#" << endl;
509     int ChosenNumber;
510     cin >> ChosenNumber;
511     while (ChosenNumber > 5 || ChosenNumber < 1)
512     {
513         cout << " 输入错误! 请重新输入! " << endl;
514         cin >> ChosenNumber;
515     }
516     for (int i = 0; i < 10; i++)
517         cout << endl;
518     switch (ChosenNumber)
519     {
520     case 1:
521         TeacherRank();
522         return true;
523     case 2:
524         SetScore();
525         return true;
526     case 3:
527         NewCourse();
528         return true;
529     case 4:
530         DeleteCourse();
531         return true;
532     case 5:
533         currentUser = nullptr;
534         return false;
535     default:
536         return false;
537     }
538 }
539
540 void UI::TeacherRank()
541 {
542     cout << setw(30) << setfill('+') << "+" << endl;
543     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
```

```
544     cout << setw(21) << setfill(' ') << " 教师成绩界面" << endl << endl;
545     cout << setw(30) << setfill('+') << "+" << endl;
546     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
547
548     cout << " 您开设的课程有: " << endl;
549     cout << setfill('-') << setw(10) << " 编号" << setw(20) << " 名称" <<
        ↪  setw(10) << " 学分" << endl;
550     for (auto item : courses)
551         if (item->GetTeacherID() == currentUser->GetID())
552             cout << setfill(' ') << setw(10) << item->GetID() << setw(20) <<
                ↪  item->GetName() << setw(10) << item->GetCredit() << endl;
553     cout << " 请输入您要显示成绩的课程号" << endl;
554     unsigned id;
555     cin >> id;
556     for (auto item : courses)
557         if (item->GetTeacherID() == currentUser->GetID() && id ==
            ↪  item->GetID())
558         {
559             auto newScore = item->GetScore();
560             for (int i = 0; i < newScore.size(); i++)
561                 for (int j = i; j < newScore.size(); j++)
562                     if (*newScore[i] < *newScore[j])
563                         swap(newScore[i], newScore[j]);
564             cout << setfill('-') << setw(10) << " 学号" << setw(10) << " 姓名"
                ↪  << setw(10) << " 成绩" << setw(10) << " 排名" << endl;
565             for (int i = 0; i < newScore.size(); i++)
566                 cout << setfill(' ') << setw(10) << newScore[i]->stuID <<
                    ↪  setw(10) << newScore[i]->stuName << setw(10) <<
                    ↪  newScore[i]->score << setw(10) << i + 1 << endl;
567             cout << endl << endl << " 请输入回车继续" << endl;
568             cin.clear();
569             cin.ignore();
570             while (cin.get() != '\n');
571             return;
572         }
573     cout << " 您输入了错误的课程号。" << endl;
574     cout << endl << endl << " 请输入回车继续" << endl;
```

```
575     cin.clear();
576     cin.ignore();
577     while (cin.get() != '\n');
578     return;
579 }
580 
581 void UI::SetScore()
582 {
583     cout << setw(30) << setfill('+') << "+" << endl;
584     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
585     cout << setw(21) << setfill(' ') << " 教师打分界面" << endl << endl;
586     cout << setw(30) << setfill('+') << "+" << endl;
587     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
588 
589     cout << " 您开设的课程有: " << endl;
590     cout << setfill('-') << setw(10) << " 编号" << setw(20) << " 名称" << endl;
591     for (auto item : courses)
592         if (item->GetTeacherID() == currentUser->GetID())
593             cout << setfill(' ') << setw(10) << item->GetID() << setw(20) <<
                ↪  item->GetName() << endl;
594     cout << " 请输入您要打分的课程号" << endl;
595     unsigned id;
596     cin >> id;
597     for (auto item : courses)
598         if (item->GetTeacherID() == currentUser->GetID() && id ==
            ↪  item->GetID())
599         {
600             cout << " 课程中的学生有: " << endl;
601             cout << setfill('-') << setw(10) << " 学号" << setw(10) << " 姓名"
                ↪  << setw(10) << " 成绩" << endl;
602             for (auto stu : item->GetScore())
603                 cout << setfill(' ') << setw(10) << stu->stuID << setw(10) <<
                    ↪  stu->stuName << setw(10) << stu->score << endl;
604             cout << " 请输入您要打分/修改成绩的学生学号: " << endl;
605             unsigned stuid;
606             cin >> stuid;
607             while (item->SetScore(stuid, 0) == false)
```

```
608             {
609                 cout << " 错误的学号! 请重新输入! (输入 0 以放弃输入) " << endl
                    ↪  << " 请输入您要打分/修改成绩的学生学号: " << endl;
610                 cin >> stuid;
611                 if (stuid == 0)return;
612             }
613             cout << " 请输入您要给的分数 (0-100): ";
614             int sc = -1;
615             cin >> sc;
616             while (sc > 100 || sc < 0)
617             {
618                 cout << " 错误的分数! 请重新输入! " << endl << " 请输入您要给的分
                    ↪  数 (0-100): ";
619                 cin >> sc;
620             }
621             item->SetScore(stuid, sc);
622             for (auto& user : users)
623             {
624                 if (user->GetType() == UserType::UnderGraduate ||
                    ↪  user->GetType() == UserType::PostGraduate)
625                 {
626                     if (user->GetID() == stuid)
627                         user->SetGrade(id, item->GetCredit(),
                            ↪  Course::GetPoint(sc));
628                 }
629             }
630             cout << " 打分完成! " << endl;
631             cout << endl << endl << " 请输入回车继续" << endl;
632             cin.clear();
633             cin.ignore();
634             while (cin.get() != '\n');
635             return;
636         }
637     cout << " 您输入了错误的课程号。" << endl;
638     cin.clear();
639     cin.ignore();
640     cout << endl << endl << " 请输入回车继续" << endl;
```

```
641     while (cin.get() != '\n');
642     return;
643 }
644 
645 void UI::NewCourse()
646 {
647     cout << setw(30) << setfill('+') << "+" << endl;
648     cout << setw(21) << setfill(' ') << " 学生管理终端" << endl << endl;
649     cout << setw(21) << setfill(' ') << " 教师开课界面" << endl << endl;
650     cout << setw(30) << setfill('+') << "+" << endl;
651     cout << setw(30) << setfill('#') << "#" << setfill(' ') << endl << endl;
652 
653     cout << " 请输入您要开设的课程名称: " << endl;
654     string newCourseName;
655     cin >> newCourseName;
656     cout << " 请输入您要开设课程的学分: " << endl;
657     int newCredit;
658     cin >> newCredit;
659     while (newCredit > 5 || newCredit < 1)
660     {
661         cin.clear();
662         cin.ignore();
663         cout << " 输入错误! 请重新输入! " << endl;
664         cin >> newCredit;
665     }
666     Course* newCourse = new Course(newCourseName, courseID++, newCredit,
        ↪  currentUser);
667     courses.push_back(newCourse);
668     cout << " 开课成功! " << endl;
669     cout << endl << endl << " 请输入回车继续" << endl;
670     cin.clear();
671     cin.ignore();
672     while (cin.get() != '\n');
673     return;
674 }
675 
676 float UI::GetJudge(unsigned teacherID)
```

```
677 {
678     for (auto item : users)
679     {
680         if (item->GetID() == teacherID)
681             return item->GetJudge();
682     }
683     return 0;
684 }
685 
686 int UI::GetRank()
687 {
688     std::vector<float> GPAList;
689     for (auto item : users)
690         if (item->GetType() == UserType::UnderGraduate)
691             GPAList.push_back(item->GetGPA());
692     sort(GPAList.begin(), GPAList.end());
693     auto rank = find(GPAList.begin(), GPAList.end(), currentUser->GetGPA());
694     return GPAList.end() - rank;
695 }
696 
697 void UI::JudgeTeacher(unsigned teacherID, unsigned judgement)
698 {
699     for (auto& item : users)
700         if (item->GetID() == teacherID)
701         {
702             item->AddJudge(judgement, currentUser->GetID());
703             return;
704         }
705 }
706 
707 void UI::SaveFile()
708 {
709     ofstream saveFile("Data.dat", ios::binary | ios::out);
710     auto userNum = users.size();
711     saveFile.write((char*)&userNum, sizeof(size_t));
712     auto courseNum = courses.size();
713     saveFile.write((char*)&courseNum, sizeof(size_t));
```

```
714     saveFile.close();
715     for (auto item : users)
716         item->SaveSelfData();
717     for (auto item : courses)
718         item->SaveSelfData();
719     return;
720 }
721 
722 void UI::LoadFile() //考虑到 STL 的问题和类的继承的问题，采用手动序列化的方式读写
723 {
724     ifstream loadFile("Data.dat", ios::binary | ios::in);
725     if (loadFile.is_open())
726     {
727         size_t userNum, courseNum;
728         loadFile.read((char*)&userNum, sizeof(size_t));
729         loadFile.read((char*)&courseNum, sizeof(size_t));
730         for (int i = 0; i < userNum; i++)
731         {
732             unsigned userID;
733             std::string userName, password;
734             size_t pwlength, nmlength;
735             UserType type;
736             loadFile.read((char*)&type, sizeof(UserType));
737             loadFile.read((char*)&pwlength, sizeof(size_t));
738             for (int i = 0; i < pwlength; i++)
739             {
740                 char newChar;
741                 loadFile.read((char*)&newChar, sizeof(char));
742                 password += newChar;
743             }
744             loadFile.read((char*)&nmlength, sizeof(size_t));
745             for (int i = 0; i < nmlength; i++)
746             {
747                 char newChar;
748                 loadFile.read((char*)&newChar, sizeof(char));
749                 userName += newChar;
750             }
```

```
751             loadFile.read((char*)&userID, sizeof(unsigned));
752             switch (type)
753             {
754             case UserType::Teacher:
755             {
756                 auto newUser = new Teacher(userID, userName, password);
757                 size_t judgeNum;
758                 loadFile.read((char*)&judgeNum, sizeof(size_t));
759                 for (int i = 0; i < judgeNum; i++)
760                 {
761                     unsigned judgement, studID;
762                     loadFile.read((char*)&studID, sizeof(unsigned));
763                     loadFile.read((char*)&judgement, sizeof(unsigned));
764                     newUser->AddJudge(judgement, studID);
765                 }
766                 users.push_back(newUser);
767                 break;
768             }
769             case UserType::UnderGraduate:
770             {
771                 auto newUser = new UnderGraduate(userID, userName, password);
772                 size_t gradeNum;
773                 loadFile.read((char*)&gradeNum, sizeof(size_t));
774                 for (int i = 0; i < gradeNum; i++)
775                 {
776                     unsigned id, credit;
777                     float point;
778                     loadFile.read((char*)&id, sizeof(unsigned));
779                     loadFile.read((char*)&credit, sizeof(unsigned));
780                     loadFile.read((char*)&point, sizeof(float));
781                     newUser->SetGrade(id, credit, point);
782                 }
783                 users.push_back(newUser);
784                 break;
785             }
786             case UserType::PostGraduate:
787             {
```

```
788                 auto newUser = new PostGraduate(userID, userName, password);
789                 size_t gradeNum;
790                 loadFile.read((char*)&gradeNum, sizeof(size_t));
791                 for (int i = 0; i < gradeNum; i++)
792                 {
793                     unsigned id, credit;
794                     float point;
795                     loadFile.read((char*)&id, sizeof(unsigned));
796                     loadFile.read((char*)&credit, sizeof(unsigned));
797                     loadFile.read((char*)&point, sizeof(float));
798                     newUser->SetGrade(id, credit, point);
799                 }
800                 users.push_back(newUser);
801                 break;
802             }
803             }
804         }
805         for (int i = 0; i < courseNum; i++)
806         {
807             size_t courseNameLength, teacherNameLength, scoreNum;
808             string courseName, teacherName;
809             unsigned CourseID, teacherID, credit;
810             loadFile.read((char*)&courseNameLength, sizeof(size_t));
811             for (int i = 0; i < courseNameLength; i++)
812             {
813                 char newChar;
814                 loadFile.read((char*)&newChar, sizeof(char));
815                 courseName += newChar;
816             }
817             loadFile.read((char*)&CourseID, sizeof(unsigned));
818             loadFile.read((char*)&teacherNameLength, sizeof(size_t));
819             for (int i = 0; i < teacherNameLength; i++)
820             {
821                 char newChar;
822                 loadFile.read((char*)&newChar, sizeof(char));
823                 teacherName += newChar;
824             }
```

```
825             loadFile.read((char*)&teacherID, sizeof(unsigned));
826             loadFile.read((char*)&credit, sizeof(unsigned));
827             loadFile.read((char*)&scoreNum, sizeof(size_t));
828             Course* newCourse = new Course(courseName, CourseID, credit,
                ↪  teacherName, teacherID);
829             for (int i = 0; i < scoreNum; i++)
830             {
831                 size_t stuNameLength;
832                 string stuName;
833                 unsigned stuID, score;
834                 float point;
835                 loadFile.read((char*)&stuNameLength, sizeof(size_t));
836                 for (int i = 0; i < stuNameLength; i++)
837                 {
838                     char newChar;
839                     loadFile.read((char*)&newChar, sizeof(char));
840                     stuName += newChar;
841                 }
842                 loadFile.read((char*)&stuID, sizeof(unsigned));
843                 loadFile.read((char*)&score, sizeof(unsigned));
844                 loadFile.read((char*)&point, sizeof(float));
845                 newCourse->AddScore(stuID, stuName, score, point);
846                 for (auto& user : users)
847                 {
848                     if (user->GetType() == UserType::UnderGraduate ||
                        ↪  user->GetType() == UserType::PostGraduate)
849                     {
850                         if (user->GetID() == stuID)
851                             user->SetGrade(CourseID, credit, point);
852                     }
853                 }
854             }
855             courses.push_back(newCourse);
856             courseID = CourseID + 1;
857         }
858         loadFile.close();
859     }
```

```
860 }
```

Course.cpp

```
1  /*
2  * 文件名: Course.cpp
3  * 用途: 完成 Score 类和 Course 类中的函数定义
4  * 作者: 2021012644 李羿璇
5  */
6
7  #include<iostream>
8  #include<vector>
9  #include<fstream>
10 #include"Course.h"
11 #include"User.h"
12
13 Course::Course(std::string nm, unsigned id, unsigned cr, User* teacher)
14 {
15     courseName = nm;
16     credit = cr;
17     courseID = id;
18     teacherID = teacher->GetID();
19     teacherName = teacher->GetName();
20 }
21
22 Course::~Course()
23 {
24     courseName.clear();
25     courseName.shrink_to_fit();
26     teacherName.clear();
27     teacherName.shrink_to_fit();
28     for (auto itr = score.begin(); itr != score.end(); itr++)
29     {
30         if ((*itr) != nullptr)
31         {
32             delete* itr;
33             *itr = nullptr;
```

```
34          }
35      }
36  }
37
38  Course::Course(std::string coursenm, unsigned courseid, unsigned cr,
    ↪  std::string teachernm, unsigned teacherid)
39  {
40      courseName = coursenm;
41      courseID = courseid;
42      credit = cr;
43      teacherName = teachernm;
44      teacherID = teacherid;
45  }
46
47  float Course::GetPoint(int sc)
48  {
49      if (sc >= 90)return 4.0;
50      if (sc >= 85 && sc < 90)return 3.6;
51      if (sc >= 80 && sc < 85)return 3.3;
52      if (sc >= 77 && sc < 80)return 3.0;
53      if (sc >= 73 && sc < 77)return 2.6;
54      if (sc >= 70 && sc < 73)return 2.3;
55      if (sc >= 67 && sc < 70)return 2.0;
56      if (sc >= 63 && sc < 67)return 1.6;
57      if (sc >= 60 && sc < 63)return 1.3;
58      return 0.0;
59  }
60
61  bool Course::SetScore(unsigned id, unsigned sc)
62  {
63      for (auto& item : score)
64      {
65          if (item->stuID == id)
66          {
67              item->score = sc;
68              item->point = GetPoint(sc);
69              return true;
```

```
70          }
71      }
72      return false;
73  }
74
75  bool Course::Quittable(unsigned studID)
76  {
77      for (auto item : score)
78      {
79          if (item->stuID == studID && item->point == -1)
80              return true;
81          else
82              continue;
83      }
84      return false;
85  }
86
87  bool Course::Judgeable(unsigned studID)
88  {
89      for (auto item : score)
90      {
91          if (item->stuID == studID && item->point != -1)
92              return true;
93      }
94      return false;
95  }
96
97  bool Course::Deletable()
98  {
99      return score.size() < 5;
100 }
101
102 void Course::QuitStudent(unsigned studID)
103 {
104     auto itr = score.begin();
105     for (; itr != score.end(); itr++)
106     {
```

```
107         if ((*itr)->stuID == studID)
108         {
109             itr = score.erase(itr);
110             return;
111         }
112     }
113 }
114 
115 void Course::AddScore(unsigned stuid, std::string stuname, unsigned sc, float
    ↪ pt)
116 {
117     Score* newScore = new Score(stuid, stuname, sc, pt);
118     score.push_back(newScore);
119     return;
120 }
121 
122 void Course::AddStudent(User* stu)
123 {
124     Score* newScore = new Score(stu);
125     score.push_back(newScore);
126 }
127 
128 void Course::SaveSelfData()
129 {
130     std::ofstream saveFile("Data.dat", std::ios::binary | std::ios::app);
131     auto courseNameLength = courseName.length(), teacherNameLength =
        ↪ teacherName.length(), scoreNum = score.size();
132     saveFile.write((char*)&courseNameLength, sizeof(size_t));
133     saveFile.write(courseName.c_str(), sizeof(char) * courseNameLength);
134     saveFile.write((char*)&courseID, sizeof(unsigned));
135     saveFile.write((char*)&teacherNameLength, sizeof(size_t));
136     saveFile.write(teacherName.c_str(), sizeof(char) * teacherNameLength);
137     saveFile.write((char*)&teacherID, sizeof(unsigned));
138     saveFile.write((char*)&credit, sizeof(unsigned));
139     saveFile.write((char*)&scoreNum, sizeof(size_t));
140     saveFile.close();
141     for (auto item : score)
```

```
142     {
143         item->SaveSelfData();
144     }
145 }
146 
147 Score::Score(User* stu)
148 {
149     stuName = stu->GetName();
150     stuID = stu->GetID();
151     score = 0;
152     point = -1;
153 }
154 
155 Score::Score(unsigned stuid, std::string stuname, unsigned sc, float pt)
156 {
157     stuID = stuid;
158     stuName = stuname;
159     score = sc;
160     point = pt;
161 }
162 
163 Score::~Score()
164 {
165     stuName.clear();
166     stuName.shrink_to_fit();
167 }
168 
169 void Score::SaveSelfData()
170 {
171     auto stuNameLength = stuName.length();
172     std::ofstream saveFile("Data.dat", std::ios::binary | std::ios::app);
173     saveFile.write((char*)&stuNameLength, sizeof(size_t));
174     saveFile.write(stuName.c_str(), stuNameLength * sizeof(char));
175     saveFile.write((char*)&stuID, sizeof(unsigned));
176     saveFile.write((char*)&score, sizeof(unsigned));
177     saveFile.write((char*)&point, sizeof(float));
178     saveFile.close();
```

```
179 }
```

User.cpp

```
1  /*
2  * 文件名: User.cpp
3  * 用途: 完成 User 及其派生类内的函数定义
4  * 作者: 2021012644 李羿璇
5  */
6
7  #include"User.h"
8  #include<iostream>
9  #include<vector>
10 #include<fstream>
11
12 User::~User()
13 {
14     password.clear();
15     password.shrink_to_fit();
16     userName.clear();
17     userName.shrink_to_fit();
18 }
19
20 void Judge::SaveSelfData()
21 {
22     std::ofstream saveFile("Data.dat", std::ios::binary | std::ios::app);
23     saveFile.write((char*)&stuID, sizeof(unsigned));
24     saveFile.write((char*)&judgement, sizeof(unsigned));
25     saveFile.close();
26 }
27
28 Teacher::~Teacher()
29 {
30     for (auto itr = judge.begin(); itr != judge.end(); itr++)
31     {
32         delete* itr;
33         *itr = nullptr;
```

```
34      }
35      judge.clear();
36      judge.shrink_to_fit();
37  }
38
39  void Teacher::SaveSelfData()
40  {
41      std::ofstream saveFile("Data.dat", std::ios::binary | std::ios::app);
42      auto myType = GetType();
43      saveFile.write((char*)&myType, sizeof(UserType));
44      auto pwlength = password.length();
45      saveFile.write((char*)&pwlength, sizeof(size_t));
46      saveFile.write(password.c_str(), sizeof(char) * pwlength);
47      auto nmlength = userName.length();
48      saveFile.write((char*)&nmlength, sizeof(size_t));
49      saveFile.write(userName.c_str(), sizeof(char) * nmlength);
50      saveFile.write((char*)&userID, sizeof(unsigned));
51      auto judgeNum = judge.size();
52      saveFile.write((char*)&judgeNum, sizeof(size_t));
53      saveFile.close();
54      for (auto item : judge)
55      {
56          item->SaveSelfData();
57      }
58  }
59
60  void Teacher::AddJudge(unsigned newJudge, unsigned studID)
61  {
62      for (auto& item : judge)
63      {
64          if (item->stuID == studID)
65          {
66              item->judgement = newJudge;
67              return;
68          }
69      }
70      auto newJudgement = new Judge(newJudge, studID);
```

```
71      judge.push_back(newJudgement);
72  }
73
74  float Teacher::GetJudge()const
75  {
76      if (judge.size() == 0)return 10;
77      unsigned total = 0;
78      for (auto item : judge)
79      {
80          total += item->judgement;
81      }
82      float result = (float)total / judge.size();
83      return result;
84  }
85
86  void GradePoint::SaveSelfData()
87  {
88      std::ofstream saveFile("Data.dat", std::ios::binary | std::ios::app);
89      saveFile.write((char*)&courseID, sizeof(unsigned));
90      saveFile.write((char*)&credit, sizeof(unsigned));
91      saveFile.write((char*)&point, sizeof(float));
92      saveFile.close();
93  }
94
95  Student::~Student()
96  {
97      for (auto itr = GPAList.begin(); itr != GPAList.end(); itr++)
98      {
99          if ((*itr) != nullptr)
100         {
101             delete* itr;
102             *itr = nullptr;
103         }
104     }
105 }
106
107 void Student::SaveSelfData()
```

```
108 {
109     std::ofstream saveFile("Data.dat", std::ios::binary | std::ios::app);
110     auto myType = GetType();
111     saveFile.write((char*)&myType, sizeof(UserType));
112     auto pwlength = password.length();
113     saveFile.write((char*)&pwlength, sizeof(size_t));
114     saveFile.write(password.c_str(), sizeof(char) * pwlength);
115     auto nmlength = userName.length();
116     saveFile.write((char*)&nmlength, sizeof(size_t));
117     saveFile.write(userName.c_str(), sizeof(char) * nmlength);
118     saveFile.write((char*)&userID, sizeof(unsigned));
119     auto gradeNum = GPAList.size();
120     saveFile.write((char*)&gradeNum, sizeof(size_t));
121     saveFile.close();
122     for (auto item : GPAList)
123     {
124         item->SaveSelfData();
125     }
126 }
127
128 void Student::QuitClass(unsigned courseID)
129 {
130     auto itr = GPAList.begin();
131     for (; itr != GPAList.end(); itr++)
132     {
133         if (courseID == (*itr)->courseID)
134         {
135             itr = GPAList.erase(itr);
136             return;
137         }
138     }
139 }
140
141 void Student::SetGrade(unsigned id, unsigned credit, float point)
142 {
143     for (auto& item : GPAList)
144     {
```

```
145         if (item->courseID == id)
146         {
147             item->point = point;
148             item->credit = credit;
149             return;
150         }
151     }
152     GradePoint* newGrade = new GradePoint(id, credit, point);
153     GPAList.push_back(newGrade);
154     return;
155 }
156 
157 unsigned Student::GetCredit()
158 {
159     unsigned totalCredit = 0;
160     for (auto item : GPAList)
161     {
162         if (item->point != -1)
163             totalCredit += item->credit;
164     }
165     return totalCredit;
166 }
167 
168 float UnderGraduate::GetGPA()const
169 {
170     int totalCredit = 0;
171     float totalPoint = 0;
172     for (auto item : GPAList)
173         if (item->point != -1)
174         {
175             totalCredit += item->credit;
176             totalPoint += item->credit * item->point;
177         }
178     float GPA;
179     if (totalCredit == 0)
180         GPA = 0;
181     else
```

```
182         GPA = totalPoint / totalCredit;
183     return GPA;
184 }
```

main.cpp

```
1  /*
2  * 文件名: main.cpp
3  * 用途: 主函数
4  * 作者: 2021012644 李羿璇
5  */
6
7  #include"Course.h"
8  #include"UI.h"
9  #include"User.h"
10
11 int main()
12 {
13     UI ui;
14     while (ui.MainWindow());
15 }
```

附录：评分表

项　　目	评　　价	
设计方案的合理性与创新性	6	
设计与调试结果	8	
设计说明书的质量	2	
程序基本要求涵盖情况	8	
程序代码编写素养情况	4	
课程设计周表现情况	2	
综合成绩	30	