项目介绍-选课系统

一．程序功能介绍

1. 账号注册与登录

用户在进入选课系统前需要拥有自己的账号，最基本的账号信息包含用户名、密码以及用户身份。注册新账号时，系统支持用户个性化地定义自己的用户名和密码，以及选择学生、教师、管理员三种身份，不同的身份在选课系统中将拥有不同的功能权限。

用户如果已经拥有账号，可以直接选择利用用户名和密码登录。不同用户的选课、开课等信息在退出系统后能得到及时记录，课程信息会在所有用户间共享。

2. 导入课程

用户身份如果是教师，会开放“导入课程”功能。在导入课程时，教师需要输入课程编号、课程类别、课程名称、课程时长、课程介绍、课程时间偏好(用于后续排课)等多个角度的信息。不同课程编号保证唯一，课程类别在设定的范围内选择，这两类信息后续会用于课程查询搜索等功能。导入的课程信息将面向所有用户开放，在课程网界面展示出来。另外，教师不仅能导入课程，也能删除自己导入的课程，灵活调整课程安排。

3. 规划课程时间

在经历导入课程这个阶段后，所有课程信息已经确定，此时可以由管理员决定排课。排课要考虑的因素包括课程种类、课程时间偏好、课程时长等，管理员可以通过课程源代码设置这些因素的权重，使得它们在排课前就已经确定了最终上课时间。

管理员在决定好参数分配后，就可以开始系统排课，最终所有课程会根据排课算法优先级确定最终的上课时间，在选课系统中公示。

4. 选课

学生在系统排课结束后，可以开始选课。学生可以通过输入课程信息的方式检索课程。选择的课程将会被自动记录到自己的课表中，在选完所有课后就能生成本学期的完整课表。学生选课也会存在一些限制，不能选择同时间的课程等(会弹出警告报错)。

5. 查询课程

用户如果希望能够快速地锁定自己希望查找的课程，可以通过以课程编号与课程类别为

索引迅速缩小符合条件的课程范围，从而实现搜索功能。

二．项目各模块与类设计细节

1、 图形界面模块:

实现的类分为工具类和图形界面类。

工具类有 Course_info、UserAuth。Course_info 用于处理与课程有关的文件读写操作，课程文件包括：总的课程信息文件“course_info.txt”、每个学生的选课课表文件“username.txt”、排课后的课程时间安排文件“alloc.txt”、每个老师适合的时间文件”prefer.txt”，Course_info 的成员函数实现了对这些文件的写入和提取需要的信息等操作。UserAuth 用于处理与账号注册登录有关的文件读写操作，用以处理记录用户账号密码的文件。

图形界面类有很多。包括初始界面 Widget，与注册登录相关联的界面类 MyRegister、Mylog，以及教师、学生、管理员三个身份使用者的主界面 CourseNetworkTeacher、CourseNetworkStudent、CourseNetworkAdmin。在 CourseNetworkTeacher 中为实现课程的导入删除操作添加了 addCourseWidget、delCourseWidget 两个界面，后期为了实现添加老师合适的时间段操作增加了 prefer 界面。在 CourseNetworkStudent 中添加了 curriculum 以实现查看课表操作，同时在学生界面通过设置一个指标 sign 来决定查看已选课程还是未选课程，减少了不必要的类的设计。在 CourseNetworkAdmin 中添加 Course 类用于实现课程分配操作，也就是下面的算法模块包装后的类。除此之外，还设计了 courseBrowserWidget 在各个界面进行使用，用于双击课程进入课程详细信息浏览界面，在这个界面中对不同身份的人开放不同的按钮操作权限，也极大的避免了设计重复的类，使用户操作更为自然。在不同界面间传参以保存身份、姓名等信息，也保证了这一操作能正常进行。

2、 排课算法模块:

在算法部分，我们使用了 Google-OR tool 作为外置的算法库，并利用其中的 Cpmodel 也即整数规划 SAT 求解器来解决这一问题。排课本质上是一个整数规划问题，因为单个课程可选的课程位置可用整数描述，所期望满足的约束条件也可用一个单个的 Objective 的最小化来描述。这一个 Objective 可以包含所有课程之间互相时间冲突的数量，可以含课程违背自己的日期倾向和时间倾向的次数，也可以将二者以某种特定权重综合起来考虑。

对于单个课程，我们采用如下的数据结构来存储（最后因时间有限尽利用了其中部分字段）

```
class Course {
public:
    int id; //课程 id，唯一字段
    std::string name;//课程名
    int hour;//课程持续时间。为简化系统，排课系统只考虑 2 课时为一节的课程。
    int school;
    int grade;
    bool is_allocated;//在排课之后变为 true
    std::vector<std::vector<int>> allocated_time;//在排课之后有值，为{day,period}
```

的二元组

```
        std::vector<bool> prefer_day; //七元组，描述日期倾向
        std::vector<bool> prefer_period; //十二元组，描述 period 倾向
        int category;
        int num; //平行课程的数量，平行课程在同一时间排课
        std::vector<int> teacher_list; //老师 ID 列表，长度与平行课程对应。

        Course(int id, const std::string& name, int hour, int school, int grade, int category,
int num, const std::vector<int>& teacher_list);
        static void allocate();
    };
```

由于排课执行的操作只有一个，这里抽象成唯一一个函数来处理：

```
void allocateCourses(std::vector<Course> &courses);
```

该函数中考虑了降低课程时间冲突数，并尽可能满足课程排在其对应的 prefer_day 和 prefer_period 中的要求。将问题转换成一个整数规划可满足性最优化问题，问题的变元为所有课程排布的时间（对于 2 小时课程来说，在只在周一到周五排课的情况下有 25 种排课时间）。问题最优化的 Objective 为：（最优化方向为最小化）

课程时间重合的对数*1 + 不在 prefer_day 中的课程数目*5 + 不在 prefer_period 中的课程数目*5

这里的 1，5，5 均为可调整的整数比例，来表示在排课需求方心中的权重。

由于要将本问题放在整数规划的体系内求解（Cpmodel 只支持极其有限的几种约束描述模式“，这里引入了一些辅助变量来表示课程是否冲突/课程是否处在 prefer 的时间段内，通过用额外的约束条件嫁接的方法来最终得到 Objective。这里我们曾做过换用 Google OR-tools 中的 MPSolver 的尝试，但 MPSolver 针对整数问题只有线性约束，这完全无法表示”bool b = (x==y)”这样的约束条件，而 Cpmodel 提供了更强大的 AddEquality 和 OnlyEnforceIf 来支持这类描述。

另一方面，这里引入外部库而非使用搜索/BP 等简单算法有以下考量：

1. 该问题本身就可以描述为整数规划，在高课程数情况下，外部整数规划库可以利用并行，显然比简单写 BP 效率更优

2. 对只考虑单独约束条件（例如课程之间的冲突），搜索在代码可读性和易写性上更高，但对多重约束条件，搜索的时间代价指数级暴增，BP 也会复杂到无法写出。

3. 整数规划虽然搭起架子来比较费劲（安装环境+学习其使用语法）比较困难，但很容易扩展各种各样的约束条件，有足够强的表达力度，在实现边际功能增量上的时间成本更低。

综合以上考量，尽管使用 OR-tools 承担了额外的代价（如花了四天时间配环境，更深一步理解了 INCLUDEPATH，LIBS，编译器版本适配等 C++文件编译的细节，花了两天卡壳最后发现要让 Qt 和 OR-tools 使用完全一致的编译器版本，理解了 linking 阶段文件顺序的重要性），但获得了更高的扩展性。

三．小组成员分工情况

毛清源、胡嘉驰：负责 QT 图形界面逻辑开发。

高国雄：负责排课算法开发。

四．项目总结与反思

QT 项目由于是多人开发，代码对接是一个比较大的问题，因为不同人写的代码可能需要不同的依赖库与编译器。在实际工作中，排课算法结合了许多 C++语法的代码，也导入了较多外部库，与QT的默认编译器存在冲突，在算法嵌入的时候需要重新配置编译器与环境，花费了很多功夫；同时排课算法需要的课程信息与最初的课程类设计也存在结构不对称，导致需要对ui界面进行多处修改。这些矛盾反映出小组分工后没有进行及时沟通交流等问题，需求没有得到统一处理，在对接效率上有待提高。

QT 自身的库其实已经实现了很多功能，并且定义了不少有帮助的类，但是在开发过程中并没有充分利用，算法部分的逻辑主要使用 C++代码实现的，导致嵌入变得很复杂。这说明在拿到任务后应该先广泛地阅读资料，找对方向与路径后再开始工作，增强对新知识的适应度与包容度。