

河南工业大学

课程报告

课程名称: 程序设计实践

专业班级: 计算机 1303 班

学生姓名: 田劲锋

学 号: 201316920311

任课教师: 唐建国

学 期: 2013-2014 学年第二学期

课程报告任务书

题目	标准化考试系统							
主要内容	<p>开发出一个标准化考试系统，所谓标准化考试系统即仅支持选择题，也是为方便自动批改的功能的实现。要求实现以下基本功能：</p> <p>(1) 提供给教师添加试题的功能（试题信息用文件保存）——输入；</p> <p>(2) 试题的整体浏览功能；</p> <p>(3) 能够抽取试题组合成一套试卷（组卷的策略：可以是随机的，当然若教师添加的试题时有知识点、章节等信息，亦可以实现按照一定的组卷策略实现出题：如每个知识点抽取若干题目，最终组合一套试卷）；</p> <p>(4) 教师实现题目信息的管理，比如删除、修改等；</p> <p>(5) 查询功能（至少一种查询方式）、排序功能（至少一种排序方式）。</p> <p>扩展功能：可以按照自己的程度进行扩展。比如(1) 简单的权限处理；(2) 成绩报表打印功能；(3) 甚至根据自己情况，可以加上学生信息和考试成绩信息的管理，并扩充为广义的考试系统。即学生输入账号密码登陆，进行考试，交卷后显示成绩；(4) 模糊查询；(5) 综合查询；(6) 统计、分析等功能。总之，可以根据自己需求进行分析功能。</p> <p>特别说明：尽可能地运用自己已经学习过的数据结构的知识去展现。</p>							
任务要求	<p>一、提交材料应包括：(1) 系统源代码(2) 课程报告</p> <p>二、整个设计过程具体要求</p> <p>(1) 需求分析 要求学生对案例系统进行分析，设计出需要完成的功能，完善各个模块的调用关系；</p> <p>(2) 设计过程 要求学生进一步明确各模块调用关系，进一步完善模块函数细节（函数名、参数、返回值等）；</p> <p>(3) 实现过程 要求学生养成良好的编码习惯、完成各个模块并进行测试，最终完成系统整体测试；</p> <p>(4) 总结阶段 按照要求完成系统设计和实现报告，并进行总结、答辩。</p>							
成绩评定	报告撰写情况（30分）			系统完成情况（30分）		答辩情况（40分）		总分
	内容 20分	规范程度 5分	程序测试 5分	基本功能 20分	扩展功能 10分	自述情况 10分	答辩情况 30分	

成绩评定教师：

目录

1	需求分析	2
1.1	任务探究	2
1.2	界面模块	2
1.2.1	学生功能模块	3
1.2.2	教师功能模块	3
2	概要设计	6
2.1	类的划分	6
2.2	List 类	7
2.3	SList 类	7
2.4	Problem 类	8
2.5	Paper 类	9
2.6	User 类	9
2.7	Score 类	10
2.8	公共回调函数	10
2.9	UI 类	12
3	详细设计	13
3.1	已实现功能	13
3.1.1	插入题目	13
3.1.2	修改题目	14
3.1.3	模糊查询	14
3.1.4	生成试卷	14
3.1.5	用户登录	14
3.1.6	学生考试	14
3.2	未实现功能	14
4	调试分析	15
5	测试结果	16
6	心得总结	17

1 需求分析

1.1 任务探究

大一最后的课程——程序设计实践，目的是加深对 C 语言的理解和使用，为后来的学习打好基础。课程设计要求做一个标准化考试系统。

任务书要求如下：

开发出一个标准化考试系统，所谓标准化考试系统即仅支持选择题，也是为方便自动批改的功能的实现。要求实现以下**基本功能**：

- (1) 提供给教师添加试题的功能（试题信息用文件保存）——输入；
- (2) 试题的整体浏览功能；
- (3) 能够抽取试题组合成一套试卷（组卷的策略：可以是随机的，当然若教师添加的试题时有知识点、章节等信息，亦可以实现按照一定的组卷策略实现出题：如每个知识点抽取若干题目，最终组合一套试卷）；
- (4) 教师实现题目信息的管理，比如删除、修改等；
- (5) 查询功能（至少一种查询方式）、排序功能（至少一种排序方式）。

扩展功能：可以按照自己的程度进行扩展。比如（1）简单的权限处理；（2）成绩报表打印功能；（3）甚至根据自己情况，可以加上学生信息和考试成绩信息的管理，并扩充为广义的考试系统。即学生输入账号密码登陆，进行考试，交卷后显示成绩；（4）模糊查询；（5）综合查询；（6）统计、分析等功能。总之，可以根据自己需求进行分析功能。

特别说明：尽可能地运用自己已经学习过的数据结构的知识去展现。

考虑基本功能，即是对题目数据库的增、删、改、查（用文件存取数据），以及按一定策略生成试卷。

考虑扩展功能，需要实现用户模块和权限处理，还有简单的成绩管理。

1.2 界面模块

对于本程序而言，我们不需要复杂的 GUI（图形用户界面）。因为标准 C 中并没有直接对图形操作的库，直接调用 Windows API 会无法移植到类 UNIX 操作系统，并且程序会相当复杂。如果可能的话，我更喜欢用 Web 的方式呈现，当然这也需要使用相应的编程工具链，不是标准 C 可以简单做到的。

所以我决定采用纯文本命令行界面，如非必要不对终端界面进行特殊设置。本来是要实现命令行参数接口的，限于时间关系，未能完成。目前的界面只是简单地文本状态下的用户交互，提供简单的容错处理。

考虑用户界面，提供学生和教师两个入口。如图 1 所示，主界面提供了两个用户入口，并提供了帮助和关于。

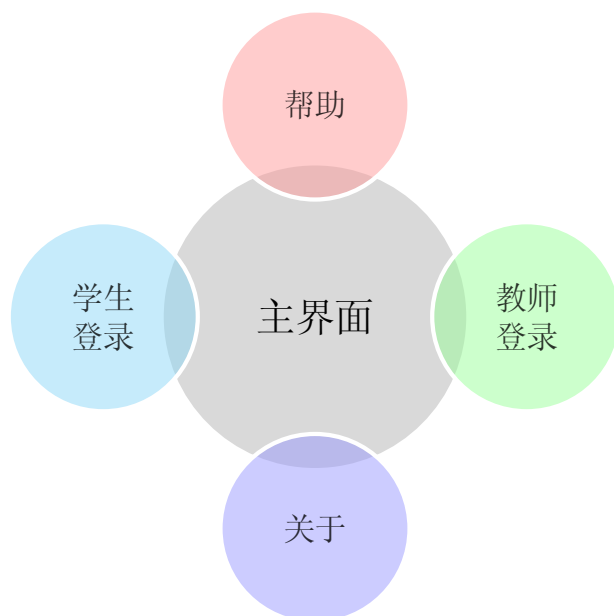


图 1: 主要模块

1.2.1 学生功能模块

学生模块属于扩展功能，如图 2，提供做卷子和看成绩两个模块入口。



图 2: 学生功能模块

1.2.2 教师功能模块

教师模块是本程序的主要部分。如图 3 所示，应该实现试题的增、删、改、查功能，实现一定策略的“智能”组卷算法。作为扩展功能，实现对成绩的查看。

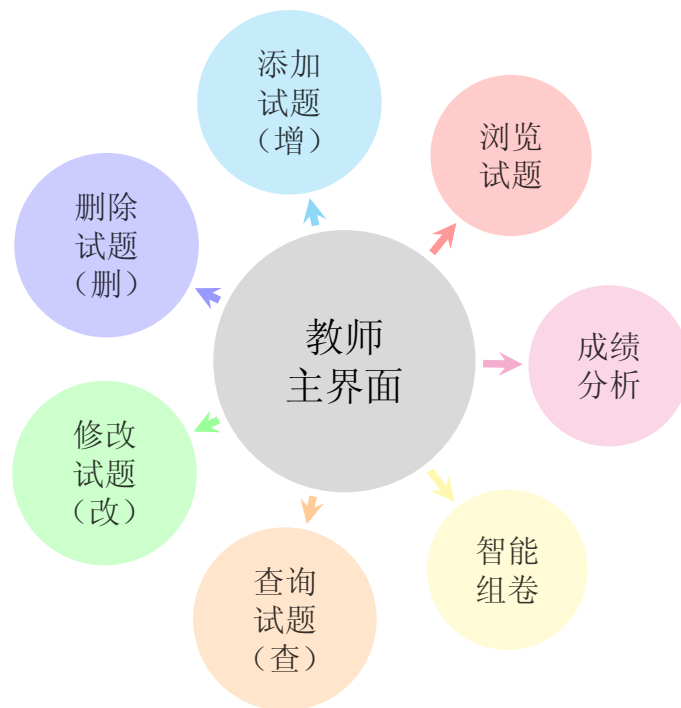


图 3: 教师功能模块

目前不需要做到批量增、删、改，其对象是应当是唯一的，这里我们用题目编号来指定。

对于查询操作，应该提供多种方式。如图 4，可以根据试题某个属性单独查询，也应提供模糊查询以查询所有选项。

对于组卷功能，我们应该提供多种算法。如图 5，可以完全随机生成，也可以指定标签、章节、难度对试卷进行生成。另外，这里也可以查看已生成试卷列表和内容。

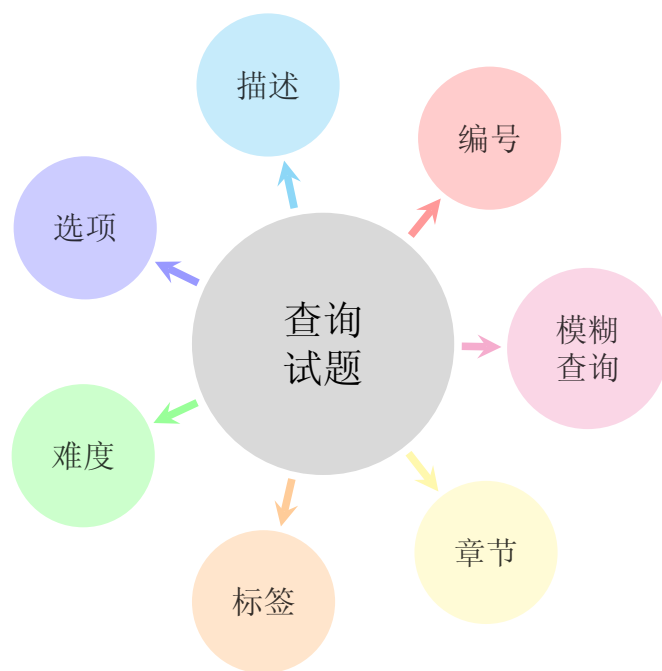


图 4: 试题查询模块

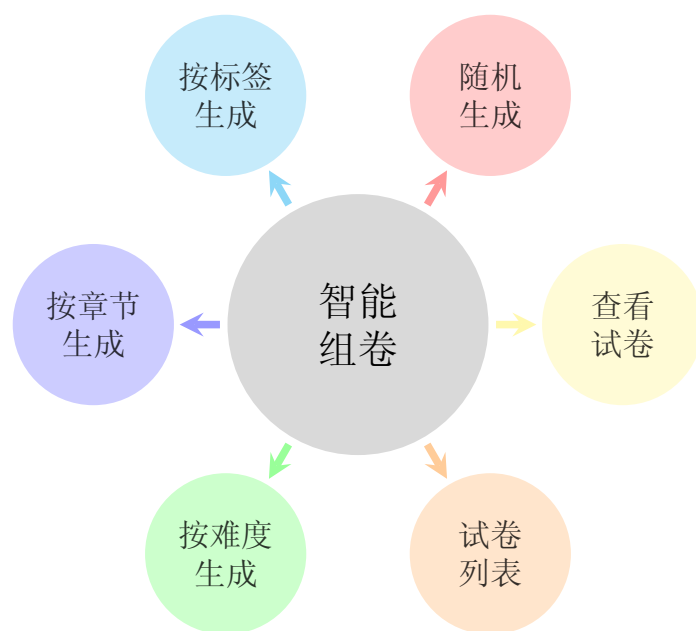


图 5: 试卷模块

2 概要设计

实际上，C 作为系统编程语言，久经考验，使用好的方法便可以写出有良好表现和维护性的代码。按传统的过程式编程方法，对于一个较大型的项目来说，有点难以驾驭。

为了程序维护和修改方便，以及程序通用性，我采用了面向对象的编程方法，参考了面向对象和设计模式的相关书籍资料¹。对于 C 语言来说，实现面向对象有点麻烦²，不过基本上还是模块化编程的思想³。

2.1 类的划分

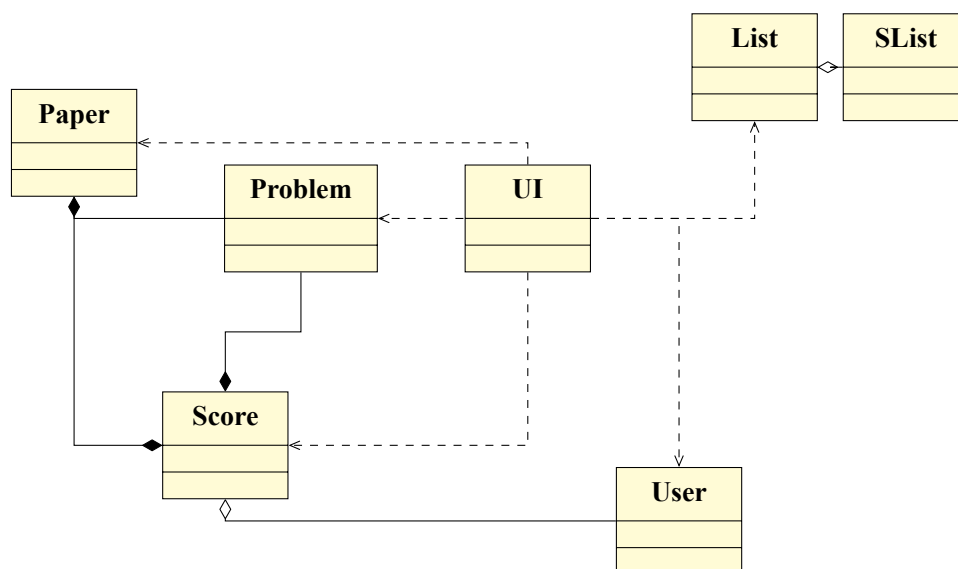


图 6: 各个类之间的关系

通过研究程序中出现的对象，我们可以抽象出若干个类。图 6 展示了各个类之间的关系。

首先，我决定使用 UI 类来控制所有与用户交互有关的操作，所有屏幕可显示字符串都存放在 UI 类中，便于修改和移植。

最重要的类是 Problem 类，提供题目存取和相关操作。Problem 类聚合可以生成试卷，即 Paper 类，用来存取试卷信息。学生和教师共用 User 类来存取，其中学生的成绩使用 Score 类存取。Score 类由 Paper 类和 User 类聚合而成，并且依赖于 Problem 类。

¹主要是《设计模式解析》[1] [2]。

²学习《系统程序员成长计划》[3] 中介绍的经验。

³参考《模块化 C：怎样编写可重用可维护的 C 语言代码》[4]。

由于没有数据库，我设计一个 List 类来实现对文件的读取、写入和增、删、改、查操作。通过 UI 类将 List 类与其他类连接起来操作（这里也是本程序设计的不佳之处）。List 类是对单链表 SList 类的一个封装。

2.2 List 类

List 类（图 7）存储一个链表，记录链表元素个数和元素中最大编号。除了初始化和销毁操作之外，提供从文件中读取、向文件中写入功能。对于链表本身，提供插入、删除、查找和遍历、查找遍历功能，通过传入回调函数来实现具体功能。

List	
- slist: SList *	//单链表头指针
+ max_id: int	//链表中元素最大编号
+ count: int	//链表中元素个数
<hr/>	
+ list_new(): List *	//创建链表
+ list_free(list: List *): void	//销毁链表
+ read_file_to_list(filename: char *, list: List *, size: size_t): int	//读取文件到链表
- list_restore(list: List *): void	//排序整理链表
# file_data_count(file: FILE *, size: size_t): int	//获取文件大小
+ write_list_to_file(filename: char *, list: List *, size: size_t): int	//写出链表到文件
+ list_insert(list: List *, p: void *): int	//插入到链表
+ list_remove(list: List *, find: 回调函数, data: void *): void *	//从列表删除
+ list_find(list: List *, find: 回调函数, data: void *): void *	//在列表中查找
+ list_each_call(list: List *, call: 回调函数, data: void *): void *	//遍历链表
+ list_find_each_call(list: List *, find: 回调, data: void *, call: 回调, userdata: void *): void *	//遍历链表中符合条件的元素

图 7: List 类

List 类实际上是对单链表 SList 类的一个封装，同时提供了文件操作，在程序中实际上是作为数据库类存在的。

2.3 SList 类

SList 类是 GNU Libltdl 库的组成部分之一，经过了长时间的考验，是非常健壮的。尊重 LGPL 协议，我未对其代码做任何改动。本来我是想改动 SList 类来实现现在 List 类的功能的，但是我还是决定尊重版权，并且提高代码重用性。

图 8 描述了 SList 类。每个 SList 对象都是一个链表元素，包含指向下一元素地址的指针和本结点元素的数据指针。

SList 类提供了链表操作的通用方法，包括附加、连接、销毁、删除、查找、

SList	
- next: <i>SList *</i>	//指向下一元素地址的指针
+ userdata: <i>const void *</i>	//指向数据域指针
+ slist_concat(head: <i>SList *</i> , tail: <i>SList *</i>): <i>SList *</i>	//链接两个单链表
+ slist_cons(item: <i>SList *</i> , slist: <i>SList *</i>): <i>SList *</i>	//插入链表元素
+ slist_delete(slist: <i>SList *</i> , delete_fct: 回调函数): <i>SList *</i>	//销毁链表
+ slist_remove(phead: <i>SList **</i> , find: 回调函数, matchdata: <i>void *</i>): <i>SList *</i>	//删除指定元素
+ slist_reverse(slist: <i>SList *</i>): <i>SList *</i>	//反转链表
- slist_sort_merge(left: <i>SList *</i> , right: <i>SList *</i> , compare: 比较函数, userdata: <i>void *</i>): <i>SList *</i>	
+ slist_sort(slist: <i>SList *</i> , compare: 比较函数, userdata: <i>void *</i>): <i>SList *</i>	//快速排序
+ slist_tail(slist: <i>SList *</i>): <i>SList *</i>	//取下一个元素
+ slist_nth(slist: <i>SList *</i> , n: <i>size_t</i>): <i>SList *</i>	//取第 <i>n</i> 个元素
+ slist_find(slist: <i>SList *</i> , find: 回调函数, matchdata: <i>void *</i>): <i>void *</i>	//查找指定元素
+ slist_length(slist: <i>SList *</i>): <i>size_t</i>	//取链表长度
+ slist_foreach(slist: <i>SList *</i> , find: 回调函数, userdata: <i>void *</i>): <i>void *</i>	//遍历每个元素
+ slist_box(userdata: <i>const void *</i>): <i>SList *</i>	//封装数据
+ slist_unbox(item: <i>SList *</i>): <i>void *</i>	//拆包数据

图 8: SList 类

遍历操作，还可以对其进行排序（使用快速排序）、反转，一般的取长度和取指定元素方法也必不可少。

2.4 Problem 类

Problem	
+ id: <i>int</i>	//题目编号
+ des: <i>char[]</i>	//题目描述
+ opt: <i>char[4][]</i>	//选项
+ ans: <i>char</i>	//答案
+ dif: <i>short</i>	//难度系数
+ tag: <i>short</i>	//知识点标签
+ chapter: <i>short</i>	//章
+ section: <i>short</i>	//节
+ problem_new(): <i>Problem *</i>	//实例化题目
+ problem_read_file(list: <i>List *</i>): <i>int</i>	//读取题目数据库
+ problem_write_file(list: <i>List *</i>): <i>int</i>	//写出题目数据库

图 9: Problem 类

图 9 所示的 Problem 类，有着题目的数据域。为了简便起见，采用四个选项的单选形式。

题目可以通过访问数据域直接操作，所以其本身的方法并不是很多，都托管给 UI 类了。

2.5 Paper 类

Paper	
+ id: <i>int</i>	// 试卷编号
+ length: <i>int</i>	// 试卷题目数
+ pid: <i>int[]</i>	// 题目编号数组
+ title: <i>char[]</i>	// 试卷标题
<hr/>	
+ papar_new(): <i>Papar *</i>	// 实例化试卷
+ papar_free(pa: <i>Paper *</i>): <i>void *</i>	// 析构试卷
+ papar_read_list(list: <i>List *</i>): <i>int</i>	// 读取试卷数据库
+ papar_write_list(list: <i>List *</i>): <i>int</i>	// 写出试卷数据库
+ paper_insert_pid(pa: <i>Paper *</i> , pid: <i>int</i>): <i>int</i>	// 插入题目
+ paper_problem_call(pa: <i>Paper *</i> , list: <i>List *</i> , call: 回调, userdata: <i>void *</i>): <i>void</i>	// 遍历试卷题目
+ paper_generate_random(pa: <i>Paper *</i> , list: <i>List *</i> , n: <i>int</i>): <i>int</i>	// 随机生成试卷
+ paper_generate_tags(pa: <i>Paper *</i> , list: <i>List *</i> , n: <i>int</i> , tags: <i>int[]</i> , m: <i>int</i>): <i>int</i>	// 按标签生成试卷
+ paper_generate_secs(pa: <i>Paper *</i> , list: <i>List *</i> , n: <i>int</i> , secs: <i>double[]</i> , m: <i>int</i>): <i>int</i>	// 按章节生成试卷
+ paper_generate_dif(pa: <i>Paper *</i> , list: <i>List *</i> , n: <i>int</i> , a: <i>int</i> , b: <i>int</i>): <i>int</i>	// 在难度区间 [a, b] 生成试卷

图 10: Paper 类

图 10 所示的 Paper 类存储试卷，用数组来存储题目数据库中的题目编号（所以试卷依赖于题目）。

在试卷中插入题目，要求题目不重复，方法会遍历题目数组保证题号唯一。生成试卷时指定的题目数量如果过多，会自动缩减到合适大小。

2.6 User 类

User	
+ id: <i>int</i>	// 用户编号
+ username: <i>char[]</i>	// 用户名
- passwd: <i>char[]</i>	// 密码
+ teacher: <i>bool</i>	// 是否为教师
<hr/>	
+ user_new(): <i>User *</i>	// 实例化用户
+ user_free(u: <i>User *</i>): <i>void *</i>	// 析构用户
- user_init(): <i>int</i>	// 初始化根用户
+ user_reg(u: <i>User *</i>): <i>int</i>	// 用户注册
+ user_login(u: <i>User *</i>): <i>int</i>	// 用户登录

图 11: User 类

图 11 是 User 类，这个类自己在内部实现了文件操作。

密码本来应该是密文存储，但是没有必要为这样一个玩具系统写 Hash 函数，不如明文存储。这里目的是展示用户登录逻辑，没有必有实现特别复杂的用户管理。

2.7 Score 类

Score	
+ id: <i>int</i>	//成绩编号
+ user_id: <i>int</i>	//用户编号
+ username: <i>char[]</i>	//用户名
+ paper_id: <i>int</i>	//试卷编号
+ paper_count: <i>int</i>	//试卷题目数
+ answer: <i>char[]</i>	//用户提交的答案
+ right: <i>int</i>	//用户题目正确个数
+ date: <i>time_t</i>	//开始做题时间
- now: <i>int</i>	//当前做到的题目指针
<hr/>	
+ score_new(<i>u: User *, p: Paper *</i>): <i>Score *</i>	//实例化成绩
+ score_read_list(<i>list: List *</i>): <i>int</i>	//读取成绩数据库
+ score_write_list(<i>list: List *</i>): <i>int</i>	//写出成绩数据库
+ score_did(<i>s: Score *, c: char, ans: char</i>): <i>int</i>	//记录做题

图 12: Score 类

图 12 是 Score 类，实现了成绩的记录。

2.8 公共回调函数

Slist 类提供了两个接口，其中回调函数接口提供了对 SList 对象的带参数调用操作，排序比较函数接口用于 SList 对象的大小比较，类似于 `qsort()` 的比较函数。

回调函数可以用于查找、输出，主要传递给 `foreach` 方法进行调用。由于传递进去的参数只有一个泛型指针，所以我设计了 `sel_num` 和 `ID`、`Block` 数据类型来存储和传递参数，具体实现比较丑陋，还有待改进。遵循此接口的方法列在图 13 中。

比较函数用于排序，目前只有按编号排序（图 14），并且隐藏在读入方法里调用。其他的关键字排序可以以此接口来编写调用，因为用途不大，不再编写。

«interface» SListCallback	
# item: <i>SList</i> *	
# userdata: <i>void</i> *	
+ write_userdata()	// 写出数据到文件
+ by_id()	// 各类编号 (查找)
+ by_des()	// 题目描述 (查找)
+ by_opt()	// 题目选项 (查找)
+ by_difr()	// 题目难度区间 (查找)
+ by_difs()	// 题目难度 (查找)
+ by_tags()	// 题目标签 (查找)
+ by_secs()	// 题目章节 (查找)
+ by_mul()	// 模糊查询 (查找)
+ by_user_id()	// 试卷中的用户编号 (查找)

图 13: 回调函数

«interface» SListCompare	
# item1: <i>SList</i> *	
# item2: <i>SList</i> *	
# userdata: <i>void</i> *	
+ cmp_id()	// 比较编号

图 14: 比较函数

2.9 UI 类

UI 类实际上相当于一个上帝类，掌控全局，并且可以访问其他类的几乎所有方法和属性。

3 详细设计

整个系统有七个大类，上百个方法。由于时间关系，不能够一一说明，我这里拣几个比较主要的方法来说明其程序流程，其他代码实现还请查看附录 ??。

3.1 已实现功能

3.1.1 插入题目

图 15 展示了插入题目的流程。

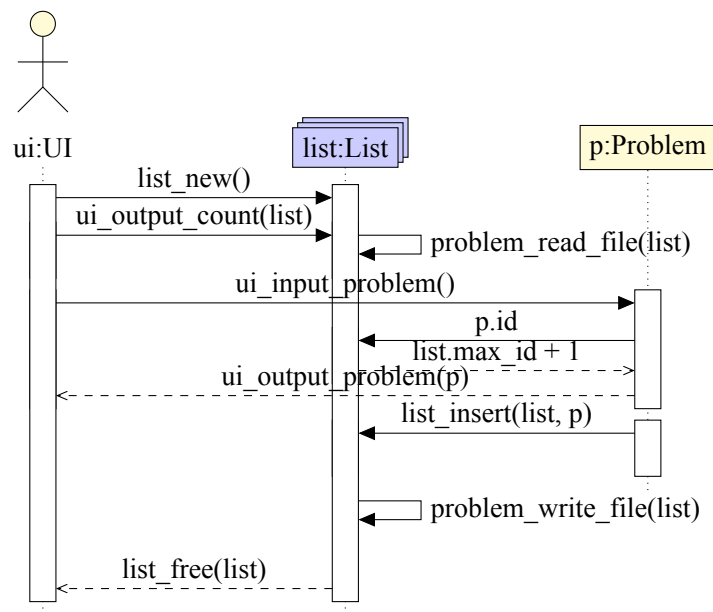


图 15: 插入题目的流程

3.1.2 修改题目

3.1.3 模糊查询

3.1.4 生成试卷

3.1.5 用户登录

3.1.6 学生考试

3.2 未实现功能

系统完成的时候，老师提出了加入多选题功能的要求。由于时间关系，不再具体实现，大概叙述一下实现的思路。

4 调试分析

5 测试结果

6 心得总结

本程序是开源的，遵循 BSD 许可证协议，全文如下：

版权所有 (c) 2014 田劲锋

保留所有权利

这份授权条款，在使用者符合以下三条件的情形下，授予使用者使用及再散播本软件包装原始码及二进制可执行形式的权利，无论此包装是否经改作皆然：

- * 对于本软件源代码的再散播，必须保留上述的版权宣告、此三条件表列，以及下述的免责声明。
- * 对于本套件二进制可执行形式的再散播，必须连带以文件以及 / 或者其他附于散播包装中的媒介方式，重制上述之版权宣告、此三条件表列，以及下述的免责声明。
- * 未获事前取得书面许可，不得使用柏克莱加州大学或本软件贡献者之名称，来为本软件之衍生物做任何表示支持、认可或推广、促销之行为。

免责声明：本软件是由加州大学董事会及本软件之贡献者以现状(as is)提供，本软件包装不负任何明示或默示之担保责任，包括但不限于就适售性以及特定目的的适用性为默示性担保。加州大学董事会及本软件之贡献者，无论任何条件、无论成因或任何责任主义、无论此责任为因合约关系、无过失责任主义或因非违约之侵权（包括过失或其他原因等）而起，对于任何因使用本软件包装所产生的任何直接性、间接性、偶发性、特殊性、惩罚性或其他任何结果的损害（包括但不限于替代商品或劳务之购用、使用损失、资料损失、利益损失、业务中断等等），不负任何责任，即在该种使用已获事前告知可能会造成此类损害的情形下亦然。

代码托管在 Github 上，可以检出到本地编译运行，网址是 <https://github.com/kingfree/haut/tree/master/clang/exam/exam>。

参考文献

- [1] A. Shalloway and J. R. Trott, *Design Patterns Explained: A New Perspective on Object-Oriented Design*. Pearson Education, 2 ed., 2005.
- [2] 徐言生, ed., 设计模式解析: 第二版 (修订版). 北京: 人民邮电出版社, 2 ed., 2013.
- [3] 李先静, 系统程序员成长计划. 北京: 人民邮电出版社, 2010.
- [4] R. Strandh, *Modular C: How to Write Reusable and Maintainable Code using the C Language*. <http://dept-info.labri.u-bordeaux.fr/~idurand/enseignement/PFS/Common/Strandh-Tutorial/reuse.pdf>, 1994.
- [5] 殷建平, 徐云, 王刚, 刘晓光, 苏明, 邹恒明, and 王宏志, eds., 算法导论 (第三版). 北京: 机械工业出版社, 3 ed., 2012.
- [6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (Third Edition)*. London, England: The MIT Press, 3 ed., 2009.
- [7] D. E. Knuth, *The Art Of Computer Programming*. Pearson Education, 1968–2011.
- [8] 高德纳, 计算机程序设计艺术. 北京: 国防工业出版社, 1992–2010.
- [9] 邓建松, 彭冉冉, and 陈长松, *L^AT_EX 2_ε 科技排版指南*. 北京: 科学出版社, 2001.
- [10] M. Kerrisk, *Linux/UNIX 系统编程手册*. 北京: 人民邮电出版社, 2014.