

火车票订票系统作业要求（v4）

2018 ACM班 数据结构

概况

实现内容

本次作业要求实现一个类似于12306的火车票订票系统，该系统包括前端、后端以及相关文档三部分。

该系统：

- 能将用户数据、购票数据、车次数据进行本地存储、并对其进行高效操作。
- 具有用户友好的操作界面，其中：
 - 能供管理员通过图形化方式管理系统，进行如增减车次，查询购票情况等操作。
 - 能供普通用户通过图形化方式使用系统，进行如注册登录，查购退票等操作。

程序结构

本次作业强制要求前后端分离，出于公平性考虑，本次作业对于前后端通信的接口进行了规定。

后端部分

后端部分在外存进行数据管理，向前端提供接口。评测的部分类似于传统OI题目，要求运行程序后进入响应输入的状态，在接受完整命令（单行命令或多行命令）后立即输出结果并再次进入响应输入的状态，直到命令要求退出系统为止。具体评测时采用管道检验输入输出是否正确。

注意：后端部分强制使用**C++**。

前端部分

前端部分调用后端，通过图形化方式向用户提供服务。调用方式，可以是作为c++编译时的一个组件；通过重定向与后端交互；或将后端构建为动态链接库并调用等等。实现的界面可以是Qt、VS等生成的，也可以是网页的形式。

注意：前端部分语言不限。

分组规则

分组采用队长招募组员的形式，队长的选择采用自行报名与指派相结合的形式，要求：

- 11个组为4人组，组内两名A班同学、两名B班同学。
- 2组为3人组，组内两名A班同学、一名B班同学。

内容分工

对于4人组，强制要求数据本地存储、修改的部分由B班同学负责，A班同学可以在思路给予帮助，但严禁借抄代码或代写等行为，希望A班同学能自觉遵守，也给B班同学一个提升的机会，code review时也会采取措施检查这一分工落实。

对于3人组，没有任何分工要求。

评测方式

性能评测（7分）

性能评测要求大家提供符合接口说明要求的后端程序，由给出的脚本进行评测。

注意：与图书管理系统相似，脚本可能多次开关用户程序，请大家做好准备。

按照程序运行速度与外存占用大小两方面进行天梯排名，并从满分递减给分。

前端评测（8分）

前端从完善度（4分）和易用度（4分）两方面进行评测，其中：

- 完善度是指前端完全用图形方式实现了整个系统能够调用到的功能。
- 易用度包含系统是否美观、布局设计是否合理、文档是否完全等方面。

注意：该项目要求提供完整的《开发文档》（包括模块划分图、每个模块的功能、类设计、文件设计等）以及完整的《使用手册》（包括系统安装手册、用户手册等），其完成情况记入易用度评分。

作业内Bonus（3分）

若你在作业中自行实现了其他有用、有趣的功能特性（不与主要要求冲突），助教组可酌情给出三分以内的Bonus，可用于弥补性能评测、前端评测中的失分。

接口说明

要求实现的命令

参数具体要求见下方数据格式部分

（特别感谢汪伟杰同学提出的诸多校对意见）

用户相关

- 用户注册

- 格式

```
register *name* *password* *email* *phone*
```

- 返回值

```
*id*( or -1)
```

- 说明

返回的id由2018开始（含2018），且2018号用户为默认管理员，每次新用户id依次递增，若注册失败则返回负数

- 样例

```
register 张三 zhangsan zhangsan@sjtu.edu.cn 12345678
```

```
-> 2018
```

- 用户登录

- 格式

```
login *id* *password*
```

- 返回值

```
0 or 1
```

- 样例

```
login 2018 zhangsan
```

```
-> 1
```

- 查询用户信息

- 格式

```
query_profile *id*
```

- 返回值

```
*name* *email* *phone* (or 0)
```

- 说明

未找到id则返回0

- 样例

```
query_profile 2018
```

```
-> 张三 zhangsan@sjtu.edu.cn 12345678
```

- 修改用户信息

- 格式

```
modify_profile *id* *name* *password* *email* *phone*
```

- 返回值

```
0 or 1
```

- 样例

```
modify_profile 2018 张三 zhangsan zhangsan@sjtu.edu.cn 87654321
```

```
-> 1
```

- 修改用户权限

- 格式

```
modify_privilege *id1* *id2* *privilege*
```

- 返回值

```
0 or 1
```

- 说明

id1 要将 id2 改为 privilege 等级

普通用户不能使用该条指令

管理员可将其他用户升级为管理员

管理员不可将管理员降级成普通用户

- 样例

```
modify_profile 2018 2018 1
```

```
->; 0
```

车票相关

- 查询车票

- 格式

```
query_ticket *loc1* *loc2* *date* *catalog*
```

- 返回值

```
listnum(\n) [train_id loc(from) date(from) time(from) loc(to) date(to) time(to)
```

```
[ticket_kind num(ticket_left) price] * num(price) (\n)]*listnum
```

```
(or -1)
```

- 说明

查询loc1到loc2发车日期为date的catalog类列车

第一行返回查到的票共有多少行，若为-1则说明查询非法；接下来每行代表一张票的相应信息。

- 样例

```
query_ticket 北京 上海 2018-03-28 CD
```

```
->; 1
```

```
c100 北京 2018-03-28 08:00 上海 2018-03-28 08:01 一等座 2000 765.50 二等座 2000 765.49 三等座  
2000 765.48
```

- 带中转查询车票

- 格式

```
query_transfer *loc1* *loc2* *date* *catalog*
```

- 返回值

```
[train_id loc(from) date(from) time(from) loc(to) date(to) time(to) dist [ticket_kind  
num(ticket_left) price] * 3 (\n)] * 2
```

```
(or -1)
```

- 说明

查询loc1到loc2发车日期为date的均为catalog类两列车，使得这两列车总时长（含中转等待时间）最短。

若有多组合法的均为最短时间中转方案，则输出第一辆列车发车更早的那一种。

第一行若为-1则说明查询非法；接下来两行代表两张可以用于中转的票务信息。

- 样例

```
query_transfer 北京 上海 2018-03-28 CD
```

```
->; c101 北京 2018-03-28 08:00 夏威夷 2018-03-28 08:01 一等座 2000 765.50 二等座 2000 765.49 三  
等座 2000 765.48
```

```
->; c102 夏威夷 2018-03-28 08:02 上海 2018-03-28 08:03 一等座 2000 765.50 二等座 2000 765.49 三  
等座 2000 765.48
```

- 订购车票

- 格式

```
buy_ticket *id* *num* *train_id* *loc1* *loc2* *date* *ticket_kind*
```

- 返回值

```
1 or 0
```

- 说明

用户id 购买 num 张 date 这天的 train_id 这辆车从loc1到loc2的ticket_kind这种票

- 样例

```
buy_ticket 666 1 C101 北京 夏威夷 2018-03-28 一等座
```

```
->; 1
```

- 查询购票信息

- 格式``

```
query_order *id* *date* *catalog*
```

- 返回值

```
listnum(\n) [train_id loc(from) date(from) time(from) loc(to) date(to) time(to) dist  
[ticket_kind num(ticket_left) price] * 3 (\n)]*listnum
```

```
(or -1)
```

- 说明

查询用户id购买的date这天类别为catalog的所有车票

第一行返回查到的票共有多少行，若为-1则说明查询非法；接下来每行代表一张票的相应信息。

- 样例

```
query_order 666 2018-03-28 C
```

```
->; c101 北京 2018-03-28 08:00 夏威夷 2018-03-28 08:01 一等座 1 765.50 二等座 0 765.49 三等座 0  
765.48
```

- 退订车票

- 格式

```
refund_ticket *id* *num* *train_id* *loc1* *loc2* *date* *ticket_kind*
```

- 返回值

```
1 or 0
```

- 说明

退订id订购的date那天train_id这列火车从loc1到loc2的ticket_kind这种票

- 样例

```
refund_ticket 666 1 C101 北京 夏威夷 2018-03-28 一等座
```

```
->; 1
```

车次相关

- 新建车次

- 格式

```
add_train *train_id* *name* *num(station)* *num(price)* *(name(price) ) * num(price)*  
*[name time(arriv) time(start) time(stopover) (price) * num(price) ] *num(station)*
```

- 返回值

```
1 or 0
```

- 说明

添加车号为train_id，名为name的车，该车经过num(station)站，共有num(price)种票价，并在之后一一列出是哪种票。

接下来的num(station)行给出各个车站相关信息。

注意刚刚新建的车次中所售车票并不能被query_ticket查询到，需要使用sale_train指令开始发售后才能被查询或购买。

- 样例

```
add_train abc123456 G123456 2 1 商务座
```

```
北京 xx:xx 08:00 00:00 ￥0.0
```

```
夏威夷 08:01 xx:xx 00:00 ￥1.5
```

```
-> 1
```

- 公开车次

- 格式

```
sale_train *train_id*
```

- 返回值

```
1 or 0
```

- 说明

将train_id的车次的车票开放发售

已发售的车次不能再发售（返回0）

- 样例

```
sale_train abc123456
```

```
-> 1
```

- 查询车次

- 格式

```
query_train *train_id*
```

- 返回值

```
train_id name num(station) num(price) (name(price) ) * num(price)
```

```
*[name time(arriv) time(start) time(stopover) (price) * num(price) ] *num(station)*
```

- 说明

查询date那天，列车号为train_id的列车

- 样例

```
query_train abc123456
```

```
-> abc123456 G123456 2 1 商务座
```

```
-> 北京 xx:xx 08:00 00:00 ¥0.0
```

```
-> 夏威夷 08:01 xx:xx 00:00 ¥1.5
```

- 删除车次

- 格式

```
delete_train *train_id*
```

- 返回值

```
1 or 0
```

- 说明

删除train_id这次列车，删除已售票的列车为非法操作

- 样例

```
delete_train abc123456
```

```
-> 1
```

- 修改车次

- 格式

```
modify_train *train_id* *name* *num(station)* *num(price)* *(name(price) ) * num(price)*
```

```
*[name time(arriv) time(start) time(stopover) (price) * num(price) ] *num(station)*
```

- 返回值

```
1 or 0
```

- 说明

修改train_id列车的信息为新信息

- 样例

```
modify_train abc123456 G123456 2 1 商务座
```

```
北京 xx:xx 08:00 00:00 ¥0.0
```

```
夏威夷 08:01 xx:xx 00:00 ¥1.5
```

```
-> 0 /* 刚才已经删掉了 */
```

管理相关

- 删库命令

- 格式

```
clean
```

- 返回值

```
1 or 0
```

- 说明

整个系统初始化，恢复到原始状态

- 关闭系统

- 格式

exit

- 返回值

BYE

数据格式

参数格式

说明：如未说带中文，则一定不带中文

- name
20B以内无空格非空字符串，带中文
- password
20B以内无空格非空字符串
- email
20B以内无空格非空字符串
- phone
20B以内无空格非空字符串
- id
20B以内无空格非空字符串
- privilege
0-2数字，0代表未注册，1代表注册用户，2代表管理员
- loc
20B以内无空格非空字符串，带中文
- date
XXXX-XX-XX
默认为X，换为数字，要求补0
- catalog
10B内大写字母非空字符串，包含所有车次类型的子集
- ticket_kind
20B以内无空格非空字符串，带中文
- train_id
20B以内无空格非空字符串
- num
整数
- time
XX:XX
默认为X，换为数字，要求补0
- price

以¥开头，后接一个浮点数

问题说明

关于用户

这次实现的用户系统仅用于存储用户数据库，并不维护“当前用户”这一概念。

关于权限

对于权限的检测应在后端的前端完成，故默认所有行为均在权限上已经过检验（即默认合法）。

关于权限的部分说明已在特定指令处给出。

未给出的部分，遵循“管理员可以进行一切操作，用户只允许操作涉及到自己的部分”这一原则。

关于车次

我们默认每天的车次是相同的，所以新建的车次和删除车次都是作用在所有日期上的。

关于车票

我们默认每种列车的每种座椅种类，初始票数均为2000。

关于管理相关命令

clean 命令用于清空数据，方便进行下一轮评测（方便删库跑路）。

exit 命令用于停止交互。

关于系统安全

在该系统中，后端信任任何发来的数据，故如果想进行安全检查以及权限检查，应在后端的前端进行。

【前端（网页或GUI）】->【后端的前端（确认通信安全可靠、确认权限、整理信息格式）】->【后端】

关于头文件限制

原则上主体功能的主要部分（如数据结构设计，中转查询算法等，不包括GUI绘制和网络通信）不应过分依赖于外部库（如stl容器），一经发现会导致成绩降低，如果实在拿捏不好程度可咨询助教组。如pthread，ctime等用于锦上添花的库的使用是支持的。

关于评测机

保证评测机（后端运行环境）至少是四核的。平台未指定（这意味着后端需要资磁多平台（不用奇奇怪怪的库问题不会很大））。

有没有什么推荐的前端啊

推荐写个网页，后端用python或者js脚本黏合一下；

推荐使用Qt，一站式解决GUI问题；

推荐开发新玩法。

自学Qt有什么好的资料吗？

中科大一个较全面的指南<https://qtguide.ustclug.org/>

一个关于网络的例子<https://blog.csdn.net/rodgerjie1993/article/details/51476456>

一些关于打包的东西<http://tieba.baidu.com/p/3730103947>

这次的命令不像上次图书馆系统一样允许参数缺省吗？

是的，这样大家parser很好写：D

为了支持新的功能，我能不能增加一些命令？

理论上可以，但是要注意新增加的命令不要和原有命令冲突（就是第一个表示什么命令的词不要一样就对啦）

前端如果只是展示的话，是不是系统任意？Windows Only也行？

是的，但是推荐使用一些跨平台的东西，比如写网页，比如Qt

如果写网络相关，去哪儿找服务器呢？

解决方案（后端放置的位置）有：

1. 本机demo, localhost
2. 使用交大宿舍ip，如果有路由器就把使用的机器设置为DMZ
3. BAT（划去）等等找个廉价的学生服务器
4. 花式使用境外小服务器

一定要写B+树吗？

所有这一组的数据结构，B/B-/B+/B*/R等等都支持。（什么？不支持哈希！）