**各模块设计说明**

**2015211313班 2015211514 吕滢 2015211516 李子茜**

1. 主模块main.cpp

接收键盘（鼠标）键入命令，分析该命令并调用相应的模块，并以时间为轴向前推进

1.1算法思想：

首先，将文件中的图书、读者数据、今天的日期读入内存，创建两个线程。

比较有特点的算法思想体现在时间轴的实现。由于不是图形化的程序，无法直接调用已有的事件驱动。因此，我们想到使用全局opration人工地进行状态的改变。

Opration初始为0。Library线程中当进入用户登录或管理员登录状态后，设置为1，当用户或管理员退出登录后重置为0。

Time线程中当opration 为1时有操作，时间轴不再继续，opration 为0时无操作，时间轴每隔10秒增加1天。

1.2相关模块：

调用user.cpp的

estab\_user\_index();读用户数据文件函数

user\_interface();用户登录界面函数

record\_user\_index();记录用户数据函数

调用books.cpp的

estab\_books\_index();读图书数据文件函数

record\_books\_index();记录图书数据函数

调用admin.cpp的

admin\_page();管理员登录界面函数

1. 图书模块 books.cpp

2.1算法思想：

刷新预约记录void books::refresh\_appointment()；

1）删除预约条目。预约过期：记录的保留起始日期比今天少三天。

2）根据馆藏数n，给前n个预约条目中 借书开始时间小于0的，置为今天。

2.2 相关模块

books.cpp的

record\_books\_index();记录图书数据函数

books::books(string set\_ID,string set\_name,string set\_author,string set\_press,string set\_isbn,int set\_time,int set\_price,int set\_total\_num);新增图书构造函数

int books::get\_rent\_num()；图书外借数量

user.cpp 的

record\_user\_index(); 记录用户数据函数

user:: user(string set\_ID, string set\_name)；新建用户的构造函数

user::user\_information()；打印读者信息

int user::get\_rent\_num()；获取用户借书数

string user:: get\_name()；获取用户名

1. 读者模块 user.cpp

3.1算法思想：

删除预约记录void user::delete\_appointment(string book\_ID)

在user类的appointment\_list中遍历到对应条目，如果找到，删除这个条目

同时删除对应book类中的这个条目

books\_map\_type ::iterator it\_book = books\_map.find(book\_ID);

books \* book\_ptr = it\_book->second;

book\_ptr->delete\_appointment(ID);

刷新预约记录void user::refresh\_appointment ()

遍历user类的appointment\_list。每找到一个条目，就刷新对应book的appointment\_list，在book的appointment\_list中找当前读者。如果找到，就把book的appointment\_list中找当前读者的预约保留日期值赋给当前user类的appointment\_list中当前书籍的预约条目；如果没找到，就把当前user类的appointment\_list中当前书籍的预约条目删除。

还书操作 void user::return\_book(string book\_ID)

先判断这本书是否被当前读者借阅，用一个局部变量记录这本书的借阅日期到，删除读者结束记录rent\_record中这本书的条目。找到这本书的变量，减少外借数量，刷新预约列表，因为多了一本书换回来，可能开始为下一位预约的读者保存。判断读者借阅是否超期，若超期罚款超期天数乘书的价格除100分位取整的金额。

借书操作void user::rent\_book(string book\_ID)

借书算法比较复杂，首先进入借书函数之前要保证读者没有达到借书上限，并且想借的书不在读者借书记录里。

函数中第一个是判断是否存在读者想借的图书，存在才可借。第二个是判断图书是否有馆藏，有馆藏才能借。这里的馆藏数，是真实的馆藏数，包括可能为预约读者保留的那些馆藏。

如果有馆藏，读者预约了并且记录，若显示这本书已经为读者保留，就删除预约的条目，成功借阅；若显示这本书没有开始为读者保留，说明预约没有轮到当前读者，继续等待。读者没有预约，若馆藏数大于预约数，成功借阅；若馆藏数小于等于预约数，无法借阅，如果没有达到预约上限，提示读者可以进行预约。

如果没有馆藏，肯定不能借阅。若读者预约了，说明预约没有轮到当前读者，继续等待；若读者预约没有预约并且没有达到预约上限，提示读者可以进行预约。

3.2相关模块

调用

books.cpp的

void books::delete\_appointment(string ID) 删除预约记录

void books::refresh\_appointment()刷新预约记录

void books::add\_appointment(string ID) 增加预约记录

void books::dec\_rent\_num()减少外借书数目

void books::inc\_rent\_num()增加外借书数目

void books::inc\_rent\_count()增加借出次数

void books::show\_information()显示图书信息

log.cpp 的

void write\_log(int type, string user\_name, string book\_ID) 填写借还书单

kmp.cpp的

string::size\_type COUNT\_KMP(const string&S,const string&T) kmp返回是否在主串中找到子串

1. 管理员模块admin.cpp

4.1算法思想：

增加图书add\_books()函数中自动生成馆藏流水号的算法

for(i=4;i<9;i++){

book\_ID\_cur[i]='0'+rand()%10 ;

}

book\_ID = book\_ID\_cur;

//查重

books\_map\_type::iterator it\_find;

it\_find = books\_map.find( book\_ID );

while(it\_find!= books\_map.end()){//编号重复

for(i=4;i<9;i++){

book\_ID\_cur[i]='0'+rand()%10 ;

}

book\_ID = book\_ID\_cur;

it\_find = books\_map.find( book\_ID );

}

先随机生成流水号，在记录中查找是否重复，直到生成唯一的流水号。

删除图书函数void admin::delete\_books()

先找到图书，再查看是否有图书在外借。

存在图书，无图书外借，即可删除图书。

删除用户函数void admin::delete\_user()

先找到用户，再查看是否仍有图书未归还。

存在用户，借书书为0，即可删除用户。

* 1. 相关模块

调用

books.cpp的

record\_books\_index();记录图书数据函数

books::books(string set\_ID,string set\_name,string set\_author,string set\_press,string set\_isbn,int set\_time,int set\_price,int set\_total\_num);新增图书构造函数

int books::get\_rent\_num()；图书外借数量

user.cpp 的

record\_user\_index(); 记录用户数据函数

user:: user(string set\_ID, string set\_name)；新建用户的构造函数

user::user\_information()；打印读者信息

int user::get\_rent\_num()；获取用户借书数

string user:: get\_name()；获取用户名

1. kmp模块 kmp.cpp

5.1算法思想：

用到了字符串匹配算法。

可以快速地找到主串中与子串相同的部分，一旦找到，就返回1，没找到返回0。

5.2 相关模块：

被读者模块的search\_by\_name()调用，用户输入的书名关键字作为子串，数据库中所有的书名作为主串，依次搜索。

1. 日志模块 log.cpp

6.1算法思想：

利用string的运算book\_ID+".txt"，生成一个文件名。

ofstream fout((book\_ID+".txt").c\_str(),ios::app);

输出流中加入模式选择ios::app可以实现在文件末尾写入，而不覆盖已有内容，可以实现日志的保存累加。

6.2相关模块：

被读者模块的user::rent\_book(string book\_ID)、user::return\_book(string book\_ID)调用，用户成功借书还书之后在书的日志文件（借书还书单）中增加条目。