**题：图书管理系统**

**一、问题描述**

H大学图书馆邀请你建立一个图书馆信息管理系统。请使用面向对象思想完成该问题，具体要求如下：

（1）设计一款**文字式交互**的图书管理系统，要求具备注册登录、浏览图书、借还图书等基本功能；

（2）要求以外部文件的形式存储书籍信息、馆藏记录、借阅记录、用户信息等。【可参考提供的书籍清单】

（3）图书信息包含但不限于：

* 题名
* ISBN/ISSN
* 作者
* 出版社
* 价格
* ……（可自行扩展）

（4）用户基类信息：

* 账户
* 密码
* 角色：1-管理员、2-读者、3-游客（可扩展）

1）管理员权限：

* 读者用户管理：支持对读者的账号进行增、删、改、查；重置读者默认密码为123456；
* 书籍信息管理：支持对馆藏图书进行增、删、改、查；

2）读者模式：

* 读者可以修改自己的密码
* 读者可以搜索图书
* 读者可以借、还图书
* 读者可以查看自己的借阅记录

（5）该系统提供根据**任一信息**搜索图书的功能：

* 题名，精确查找到书
* ISBN/ISSN，精确查找到书
* 作者，模糊查找到该作者所有书，字典序排序
* 出版社，模糊查找到该出版社所有书，字典序排序。
* 有多个查询结果显示时，请按字典排序分页显示
* **搜索不考察性能，仅考察功能**

（6）该系统提供排行榜功能

* 借阅次数前十的图书
* 借阅次数前十的读者
* 最新出版前十的图书
* ……（可扩展）

（7）该系统应有明确的分级菜单显示和跳转功能，可以从上级菜单进入下级菜单，也可以从下级菜单返回上级菜单

**二、分析设计**

**数据结构的定义**

1. **类设计**
2. library类：采用对象组合的方式，library类中含有book、admin、user、record的vector容器，且library类是其他类的友元，可访问私有数据
3. book类：单本书籍，user类的友元，可在借还书时修改user的借书数据

数据包括基础数据和借阅数据，借阅数据：

借阅状态int state 值1为借阅中，0为未借阅，默认未借阅

借阅记录ID vector<int>recordId 通过ID查找索引至借阅记录，大小为借阅次数

1. admin类：单个管理员，数据包括基础数据
2. user类：单个读者

数据包括基础数据和借阅数据，借阅数据：

当前借阅书籍数int bookNum，默认为0

借阅记录ID vector<int>recordId 通过ID查找索引至借阅记录，大小为借阅次数

1. record类：单条记录

数据包括基础数据：

借阅状态int state 值1为借阅中，0为已归还

借阅记录ID int id

书籍ID和读者ID int bookId,userId 均为索引至book类和user类的数据，便于显示和修改信息

借阅时间和归还时间 time\_t timeBorrow,timeBack

**数据补充说明：**ID的主要用途是查找索引其他类中的对象，以显示和修改信息。用储存指针或者是vector的迭代器的方法来实现效率更高，免去查找的过程。但由于vector数组的erase操作会导致删除对象之后的所有对象前移，原来储存的指针失效或者迭代器错位，尚未找到合适的解决方法，所以用储存ID的方法来找到对应的记录、书籍、用户。

**程序主体结构设计**

**1.项目结构**

Codeblocks自动生成的项目结构，其中include文件夹存放类的声明，src文件夹存放类的定义，主文件夹存放main函数，自创建data文件夹存放外部txt数据

**2.main函数**

创建library类对象H，调用H.land()进入登录界面

**函数设计、参数规格以及返回类型**

**全局函数**

**1.vector<string> split(const string& s, char sep)**

封装好的以特定字符sep分隔整行字符串s，用于读入文件，避免书名等数据中含有空格影响读入，返回分隔好的vector<string>容器。

**2.void getTime()**

获取当前时间，格式化输出，作为刷新的标志。

**3.void getTime(time\_t &timep)**

将给定时间timep格式化输出，用于显示借阅时间和归还时间。

**library类**

**4.library::library()**

library类的构造函数，调用

bookInitial();userInitial();adminInitial();recordInitial();以读取文件初始化library类。

**5.void library::bookInitial()**

book初始化读入文件，stringstream和split函数结合读入每行数据，创建临时对象并调用book构造函数，并通过vector的push\_back函数拷贝构造添加至hbook容器的末尾。

**6.void library::userInitial()**

user初始化读入文件，类似如上。

**7.void library::adminInitial()**

admin初始化读入文件，类似如上。

**8.void library::recordInitial()**

record初始化读入文件，类似如上，同时将record中的数据查找添加至对应的user和book中。由于是初始化，可以利用vector容器的顺序，直接通过访问下标添加记录。但如果有删除的情况，顺序可能和id不符，所以再用find，保证正确性的同时加快效率。

**9.void library::add(const user& thisUser)**

user添加写入文件，直接将传递的参数添加至文件的末尾。

**10.void library::add(const admin& thisAdmin)**

admin添加写入文件。

**11.void library::add(const book& thisBook)**

book添加写入文件。

**12.void library::add(const record& thisRecord)**

record添加写入文件。

**13.void library::changeUser()**

user修改写入文件，由于位置不确定，直接将所有内容覆盖重写。

**14.void library::changeBook()**

book修改写入文件。

**15.void library::changeRecord()**

record修改写入文件。

**16.void library::land()**

登录界面，通过switch判断调用enroll(),adminUI(),userUI(),visitorUI()函数以跳转界面。登录成功的判断用find函数，返回迭代器，其中find的第三个参数需要重载运算符以匹配对应的类型。以下此代码所有查找功能都使用了find函数返回迭代器+重载运算符实现，不再重复说明。

**17.void library::enroll()**

注册界面，用户名查重+vector添加元素。

**18.void library::adminUI()**

管理员界面，调用manageUser(),manageBook()跳转至管理读者和书籍界面，调用searching相应的重载函数显示所有信息。

**19.void library::manageUser()**

读者管理界面，包含增加，查找，修改，删除功能。

**20.void library::manageBook()**

书籍管理界面，包含增加，查找，修改，删除，排行榜功能，调用searchingUI和rankingUI，传入管理员type参数以在相同UI实现特定功能。

**21.void library::userUI(int id)**

用户界面，包含搜索书籍，查看个人借阅，查看排行榜，修改密码功能，调用searchingUI和rankingUI，传入用户type参数以在相同UI实现不同功能，传入id以在对应函数借阅；查看个人借阅，调用book::rlend(vector<user>::iterator &thisUser)函数，返回值int类型，判断返回值若为1则归还，否则不归还。

**22.void library::visitorUI()**

游客界面，包含搜索书籍和查看排行榜功能，调用searchingUI和rankingUI，传入游客type参数以在相同UI实现特定功能。

**23.void library::searchingUI(int type,int id)**

搜索界面，包含精确搜索、模糊搜索和显示所有书籍，精确搜索在此界面中，模糊搜索和全部书籍用sort函数加lambda表达式自定义比较函数字典序排序并传参调用searching，type的值对应角色类型，为方便操作管理员可于此修改图书，调用book::revise()，用户可于此借阅图书，调用book::lend(vector<user>::iterator &thisUser)，游客无法有多余操作。

**24.void library::searching(vector<book\*>&result,int type,int id)**

模糊搜索或者显示所有书籍，通过下标和while循环跳转页数，type的值对应角色类型，为方便操作管理员可于此修改图书，用户可于此借阅图书，游客无法有多余操作。

**25.void library::searching(vector<record\*>&result)**

显示所有借阅记录。

**26.void library::searching(vector<user\*>&result)**

显示所有用户。

**27.void library::searching(vector<admin\*>&result)**

显示所有管理员。

**28.void library::rankingUI(int type,int id)**

排行榜界面，选择，排序，调用ranking函数，同searchingUI函数的调用方法。

**29.void library::rankingUI(int type,int id)**

显示图书排行，同searching函数的角色判断。

**30.void library::ranking(vector<user\*>&result,int type,int id)**

显示读者排行。

**admin类**

**31.void admin::view()**

单个管理员信息展示。

**record类**

**32.void record::userView()**

单条记录信息展示，读者查看模式，省略用户ID。

**33.void record::adminView()**

单条记录信息展示，管理员查看模式，显示用户ID。

**user类**

**34.void user::view()**

单个读者信息展示，用于排行榜中的显示，不能显示密码。

**35.void user::privateView()**

单个读者信息展示，管理员查看模式，可显示密码。

**book类**

**36.void book::view()**

单本书信息展示。

**37.int book::lend(vector<user>::iterator &thisUser)**

书籍借阅，已借阅或每人满3本正在借阅或取消借阅返回0，借阅成功返回1。

**38.int book::rlend(vector<user>::iterator &thisUser)**

个人借阅记录详细书籍信息查看，同时判断是否已归还，若未归还，归还成功返回1，已归还或归还失败返回0。

**39.int book::revise()**

书籍信息修改或删除，同时判断是否已删除，若已删除返回1，未删除返回0。

**函数补充说明：**各类中的运算符重载和构造函数为内联函数，不详细说明；省略添加和修改信息时调用add函数和changeBook等函数的详细说明；每个界面函数都有while循环和switch结合，避免了goto的使用同时能跳转上下级以及及时用system(“pause”)和system(“cls”)清屏，且输入错误都可以重新输入或者退回上一级。

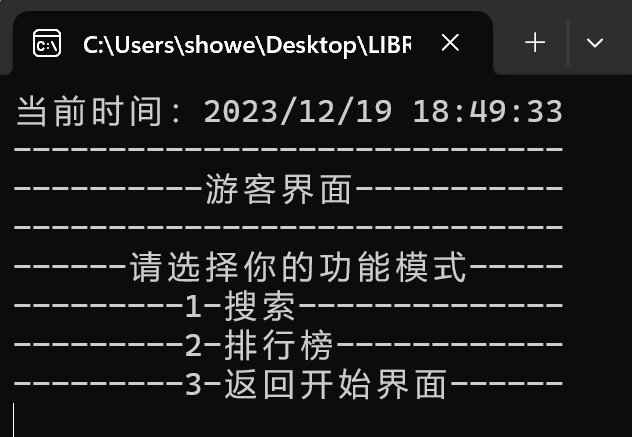
**三、输入输出测试**

**1.开始界面**



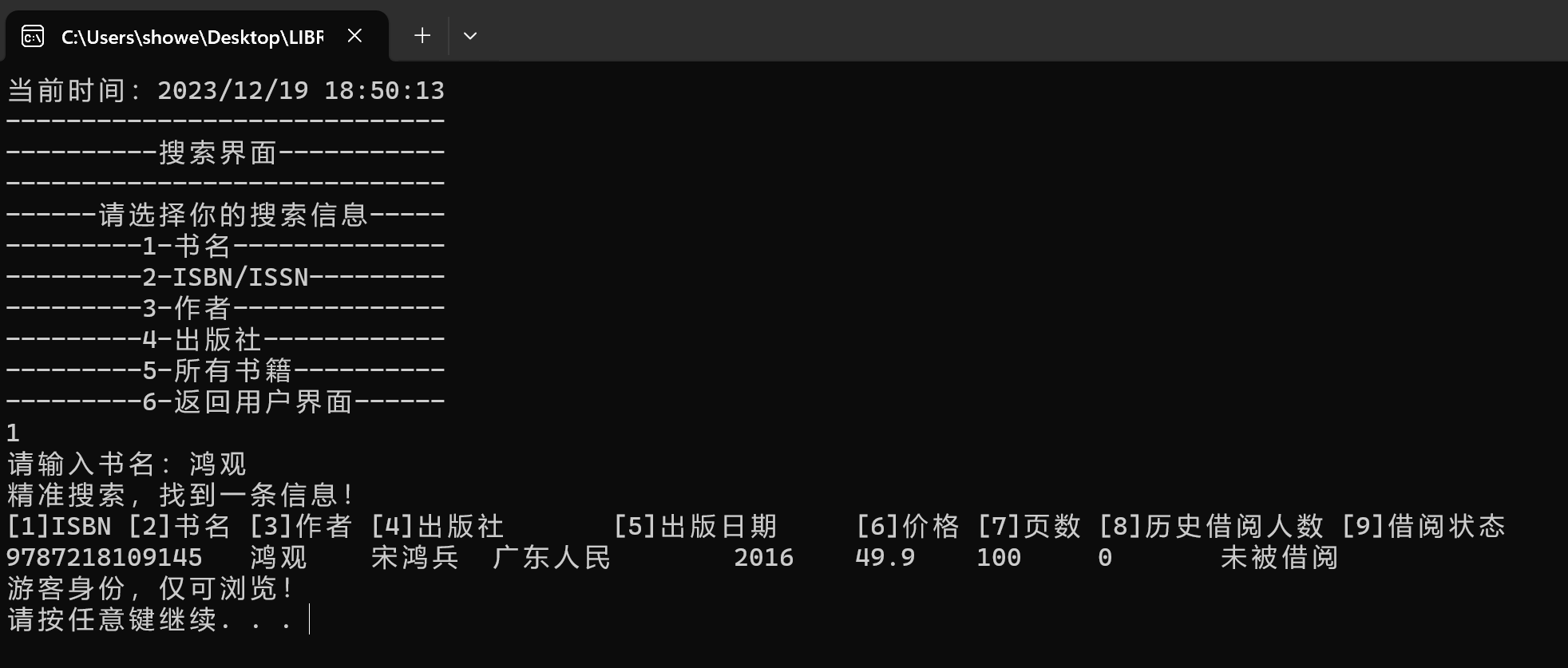
**2.游客模式**

**游客界面**

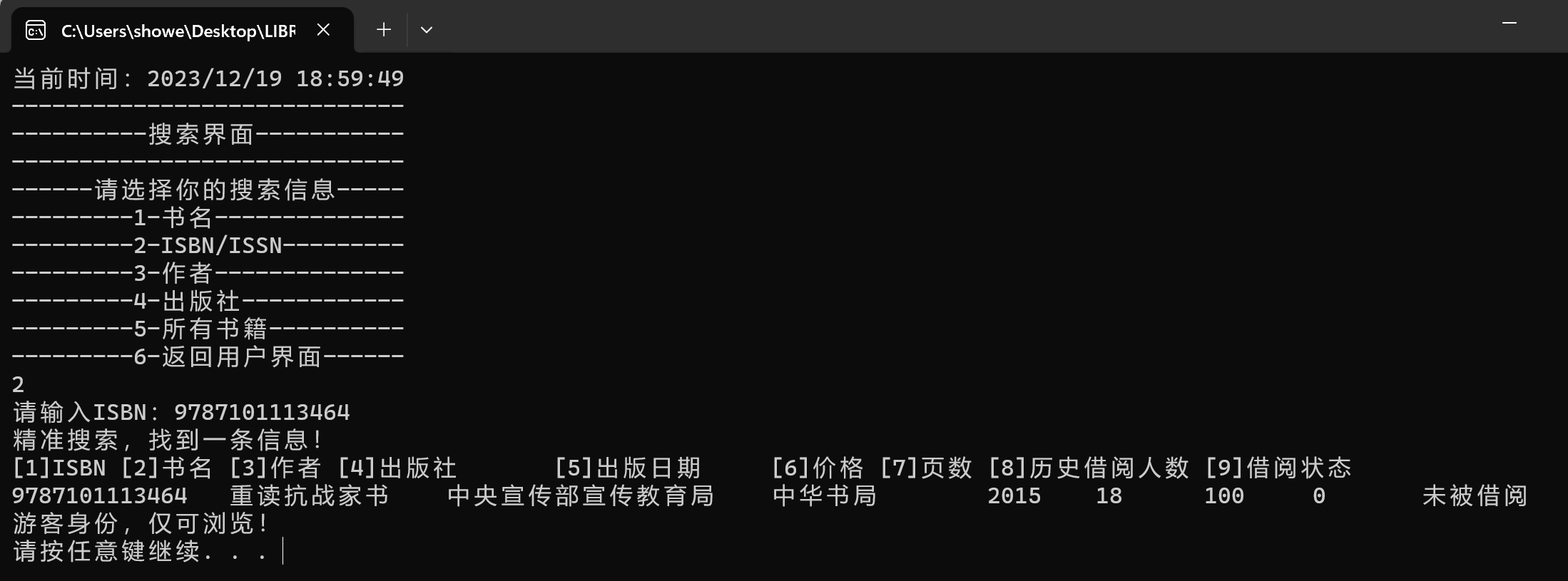


**搜索界面**（其他角色搜索界面相似，区别仅为管理员可修改或删除搜索到的书籍，读者可借阅搜索到的书籍）

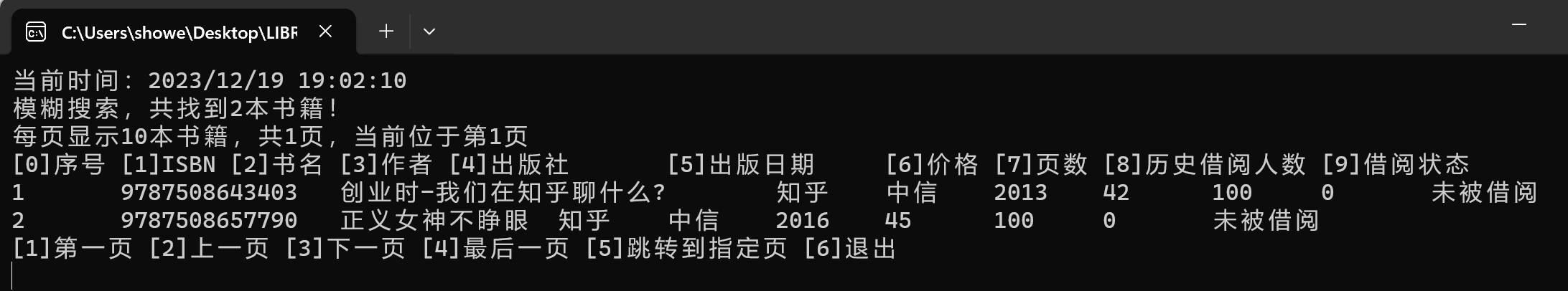
精确搜索：书名



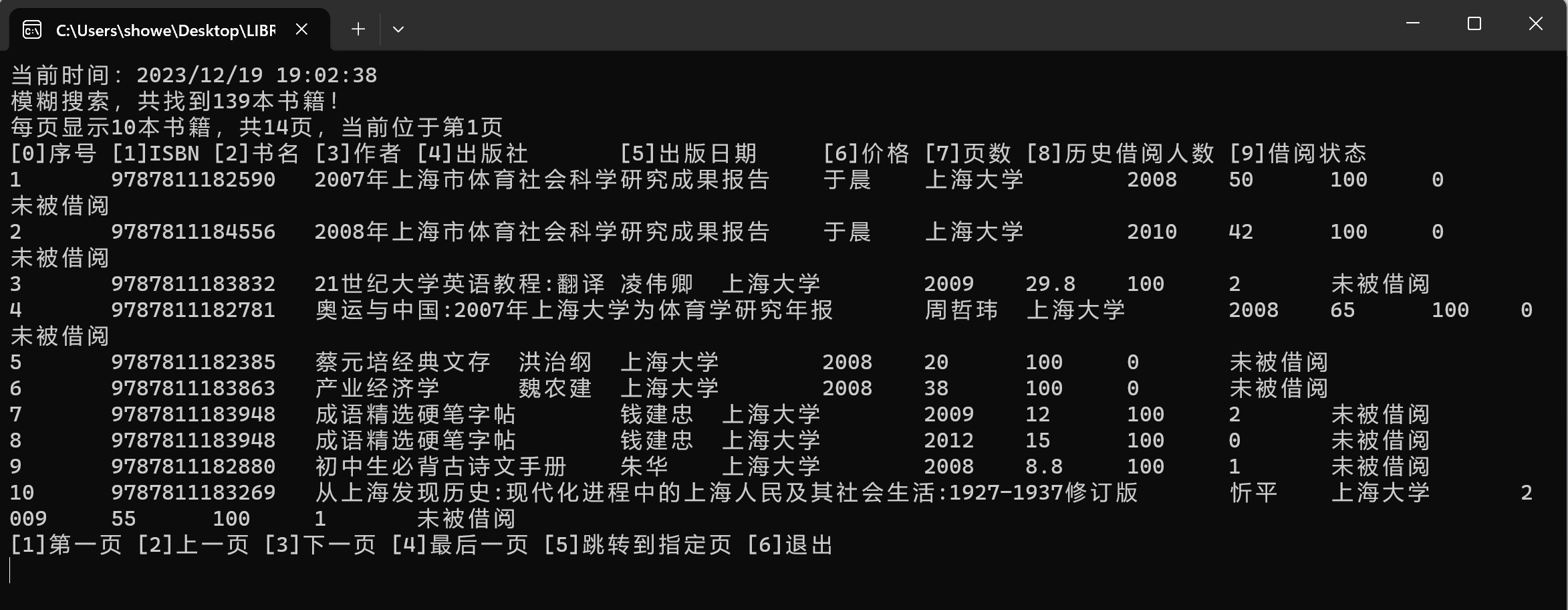
精确搜索：ISBN



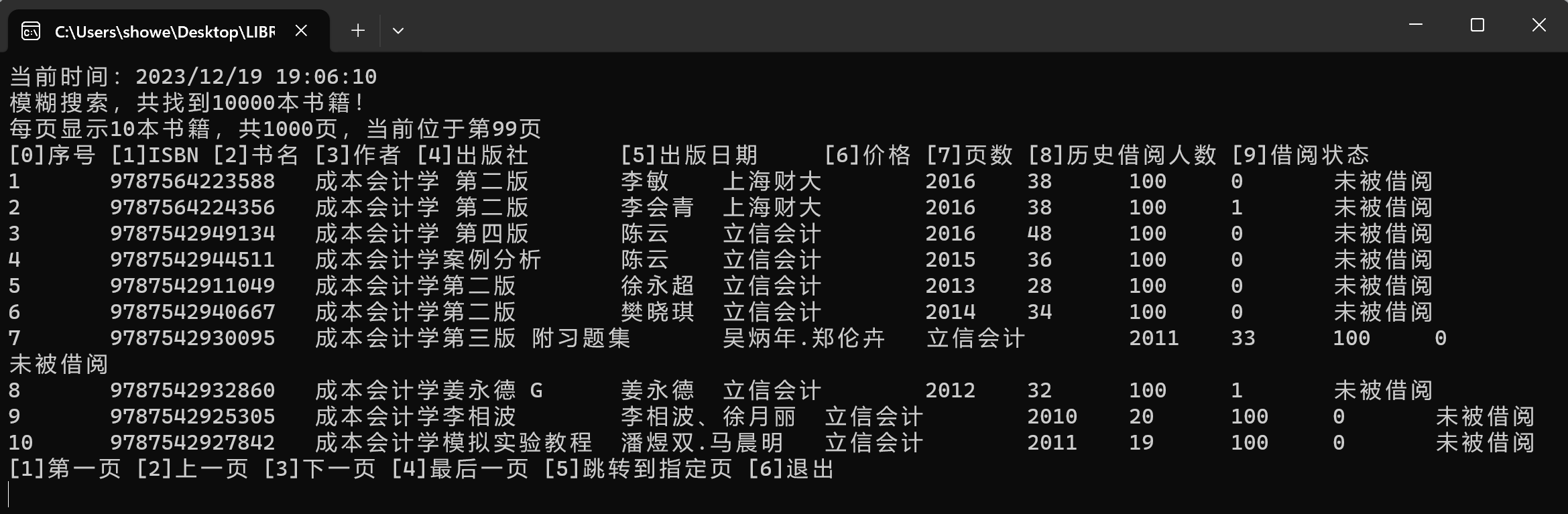
模糊搜索：作者（支持第一页、上一页、下一页、最后一页、跳转至指定页和返回，多页显示以字典序排序）



模糊搜索：出版社

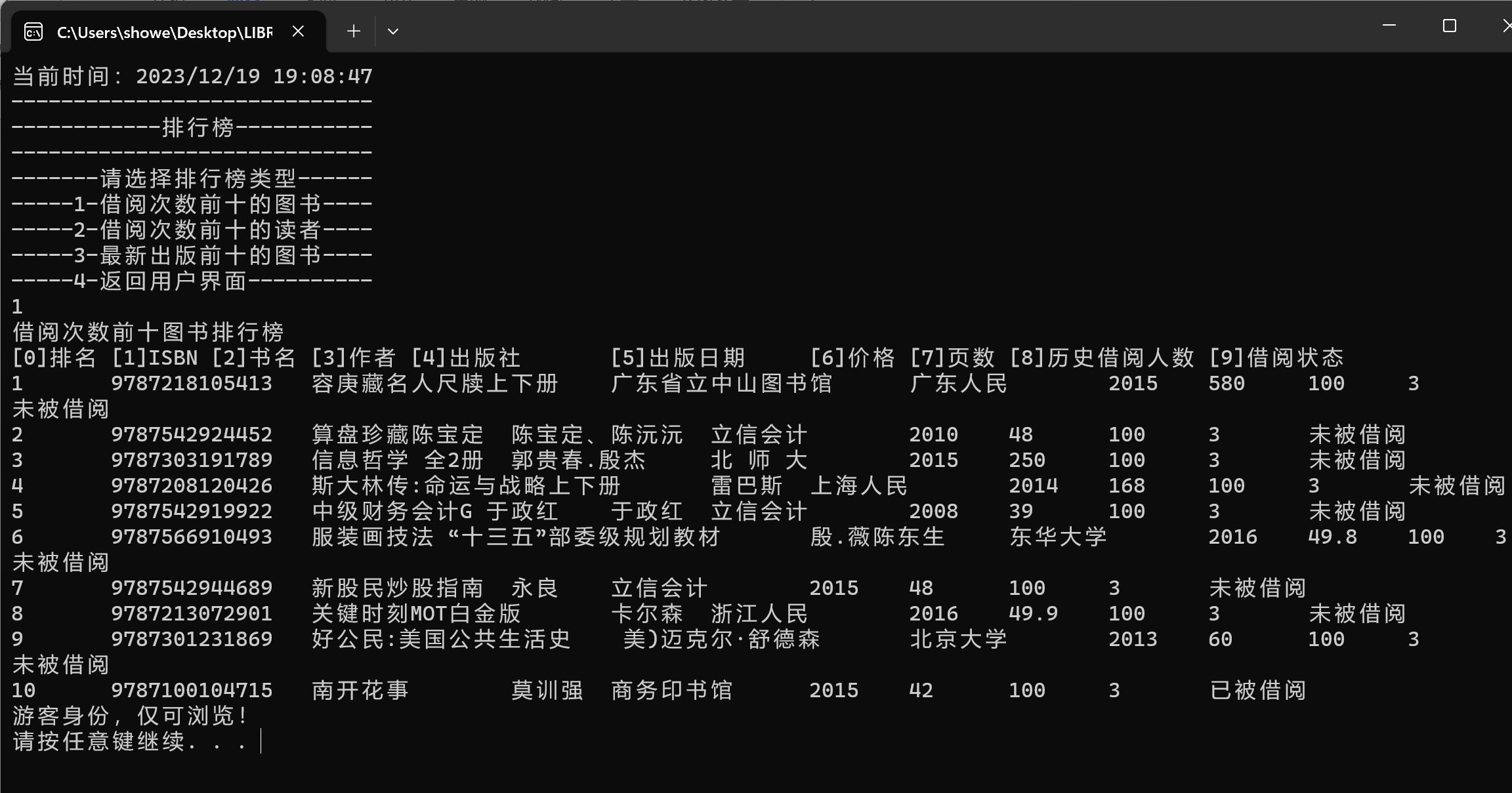


全部书籍展示（跳转至99页的展示）**已导入10000本书数据**



**排行榜**（其他角色排行榜页面类似，区别仅为管理员可修改或删除书籍，读者可借阅书籍）

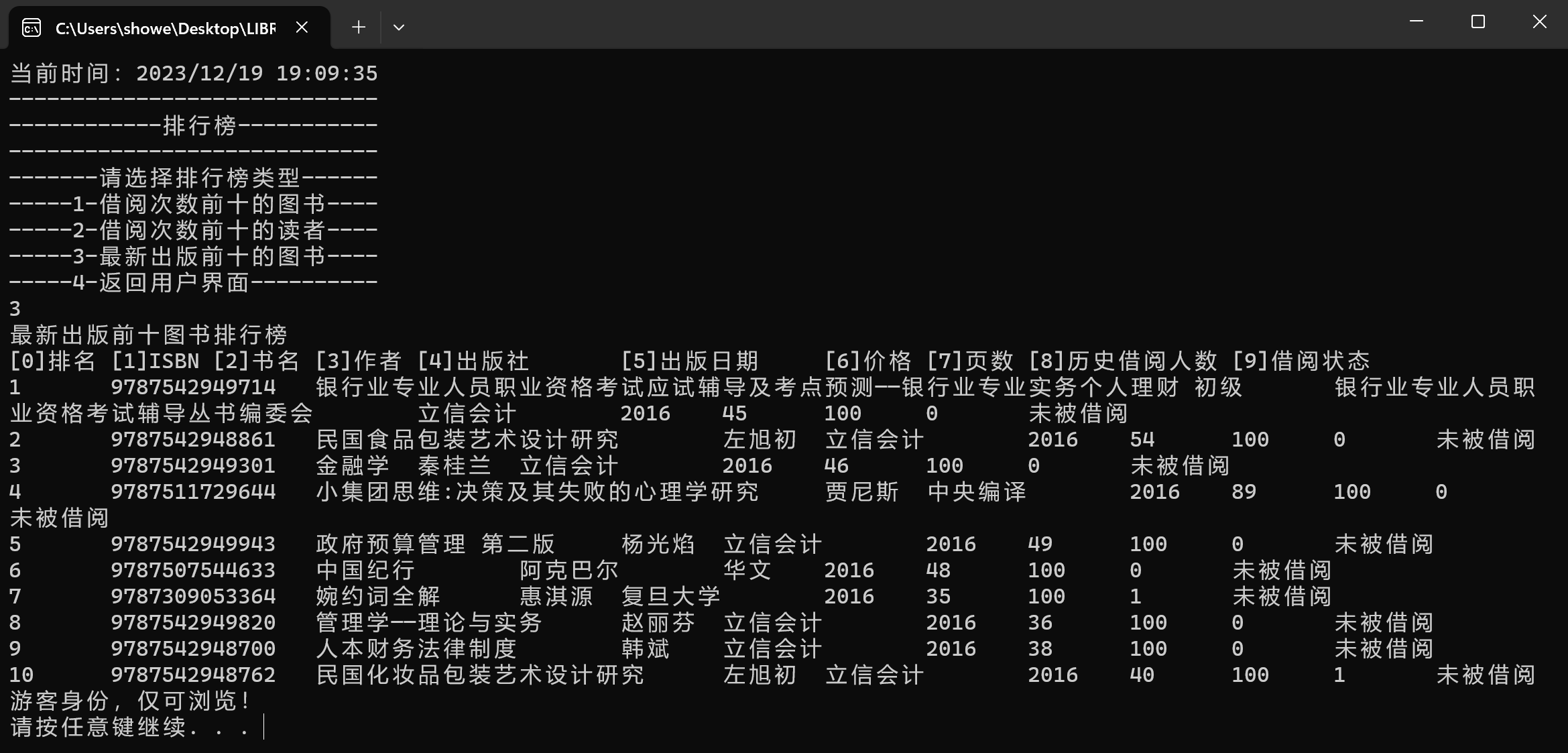
排行榜类型1 借阅次数前十图书



排行榜类型2 借阅次数前十读者

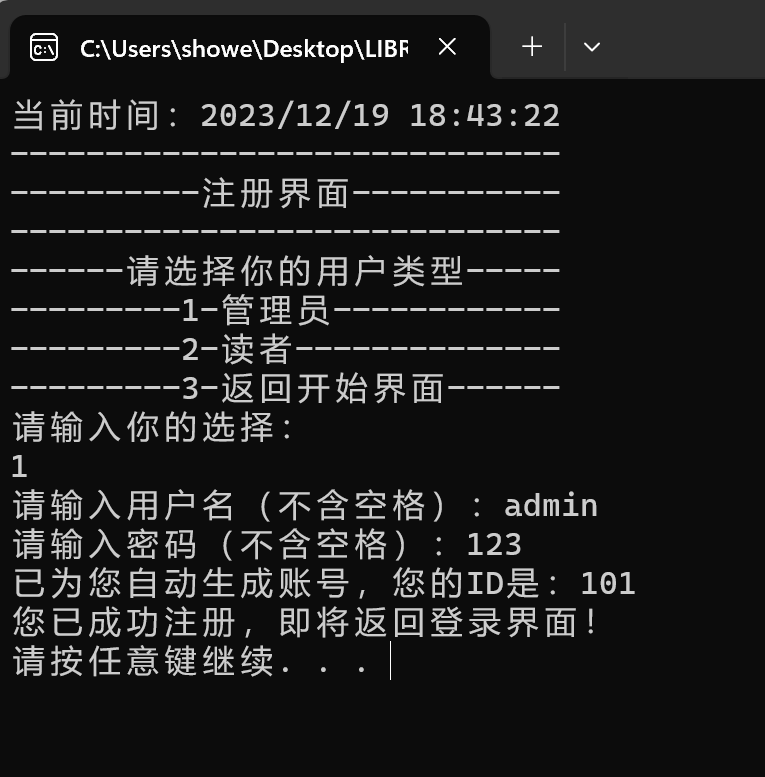


排行榜类型3 最新出版前十图书

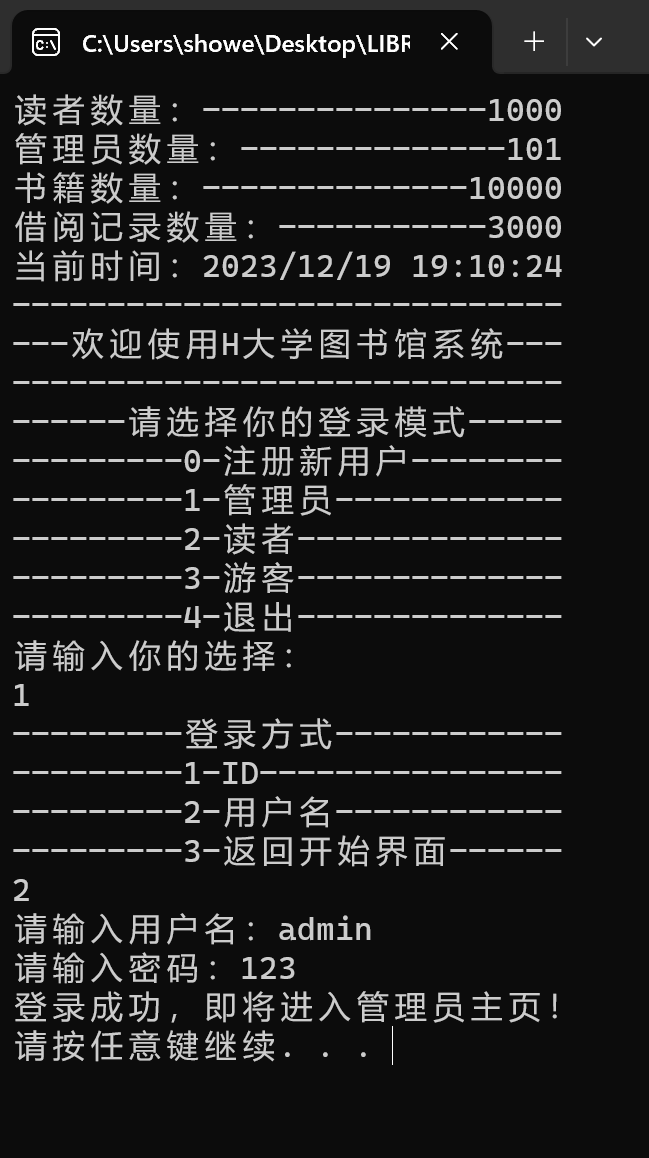


**3.管理员模式**

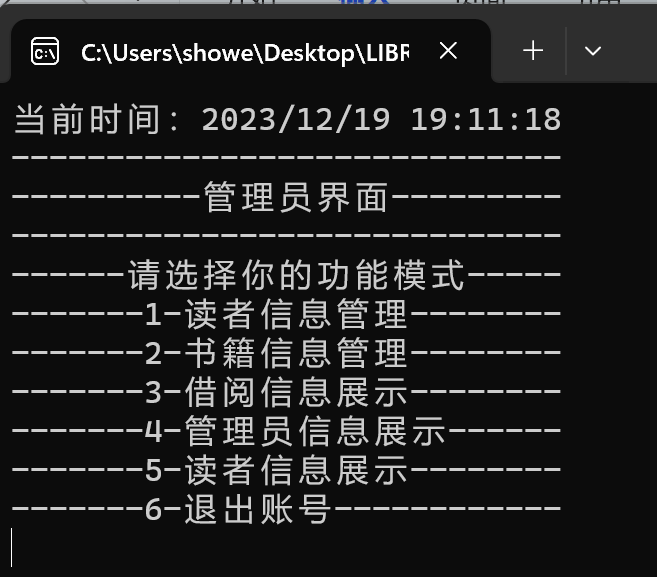
**注册界面**



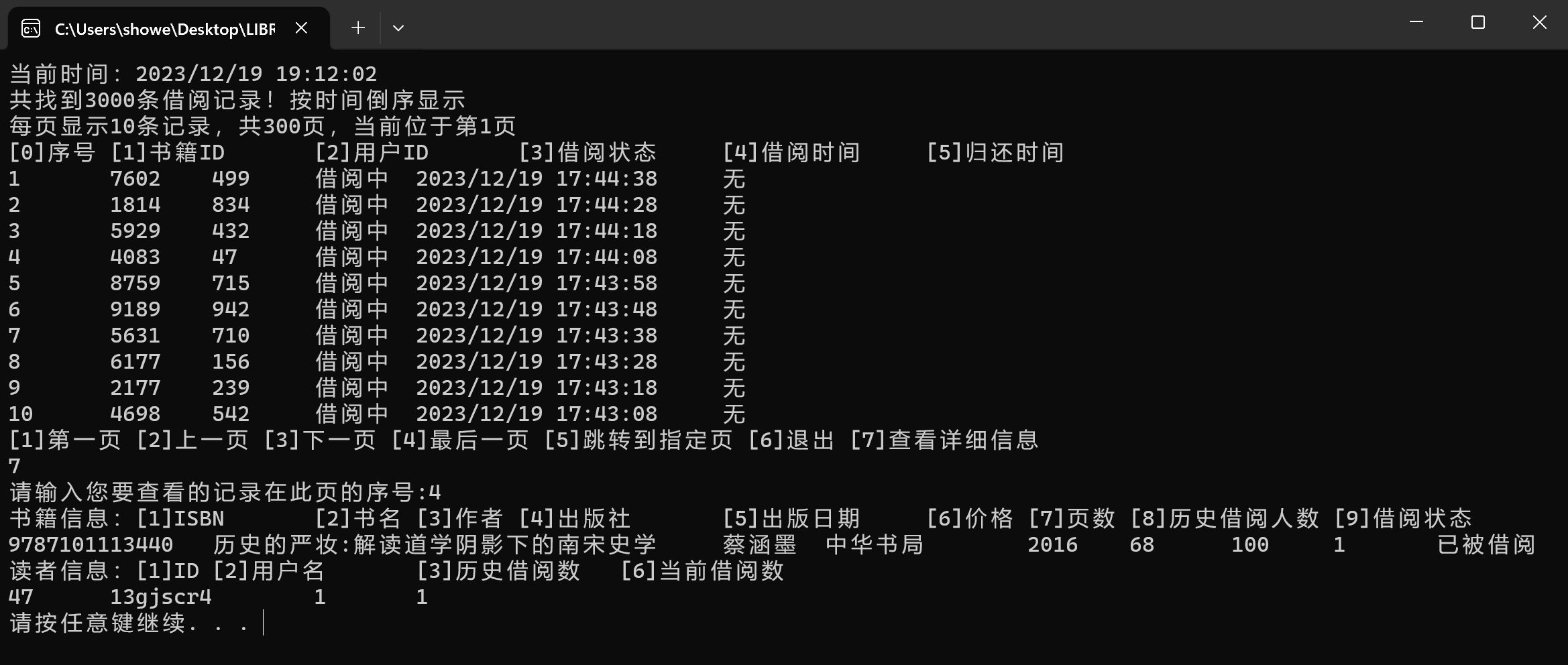
**登录界面**



**管理员界面**



全部借阅信息展示 **已导入3000条**



全部管理员信息展示 **已导入100名**



全部读者信息展示 **已导入1000名**

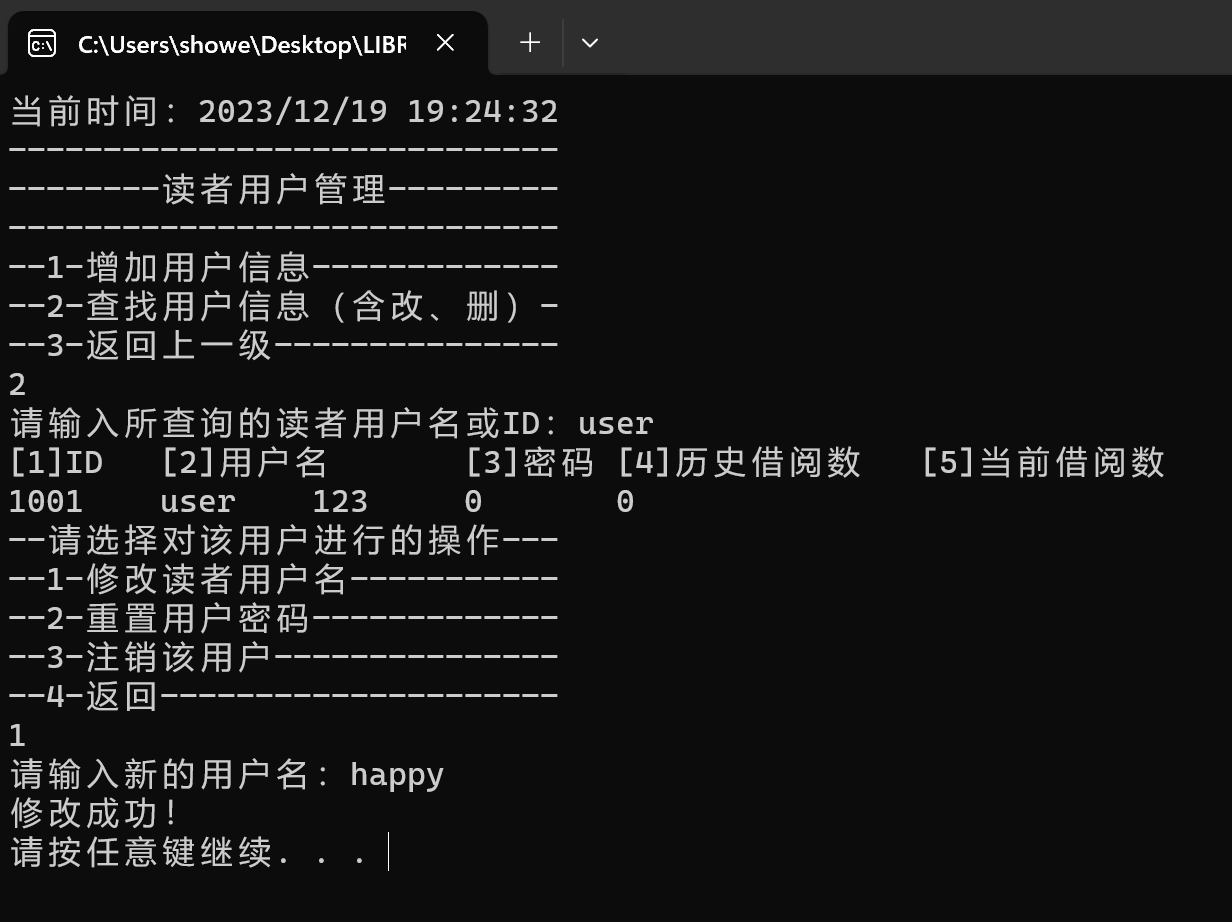


**读者用户管理**

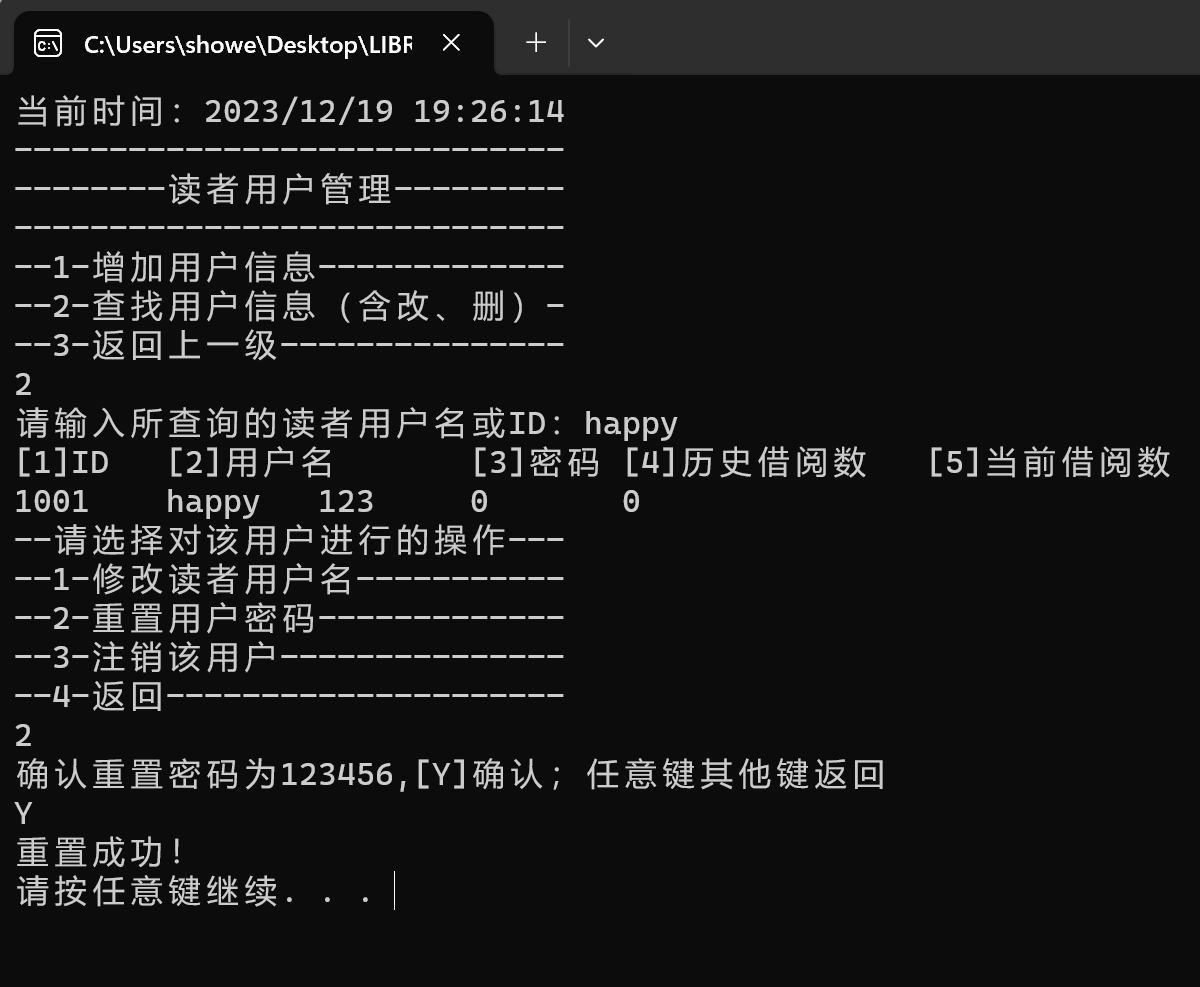
增添信息



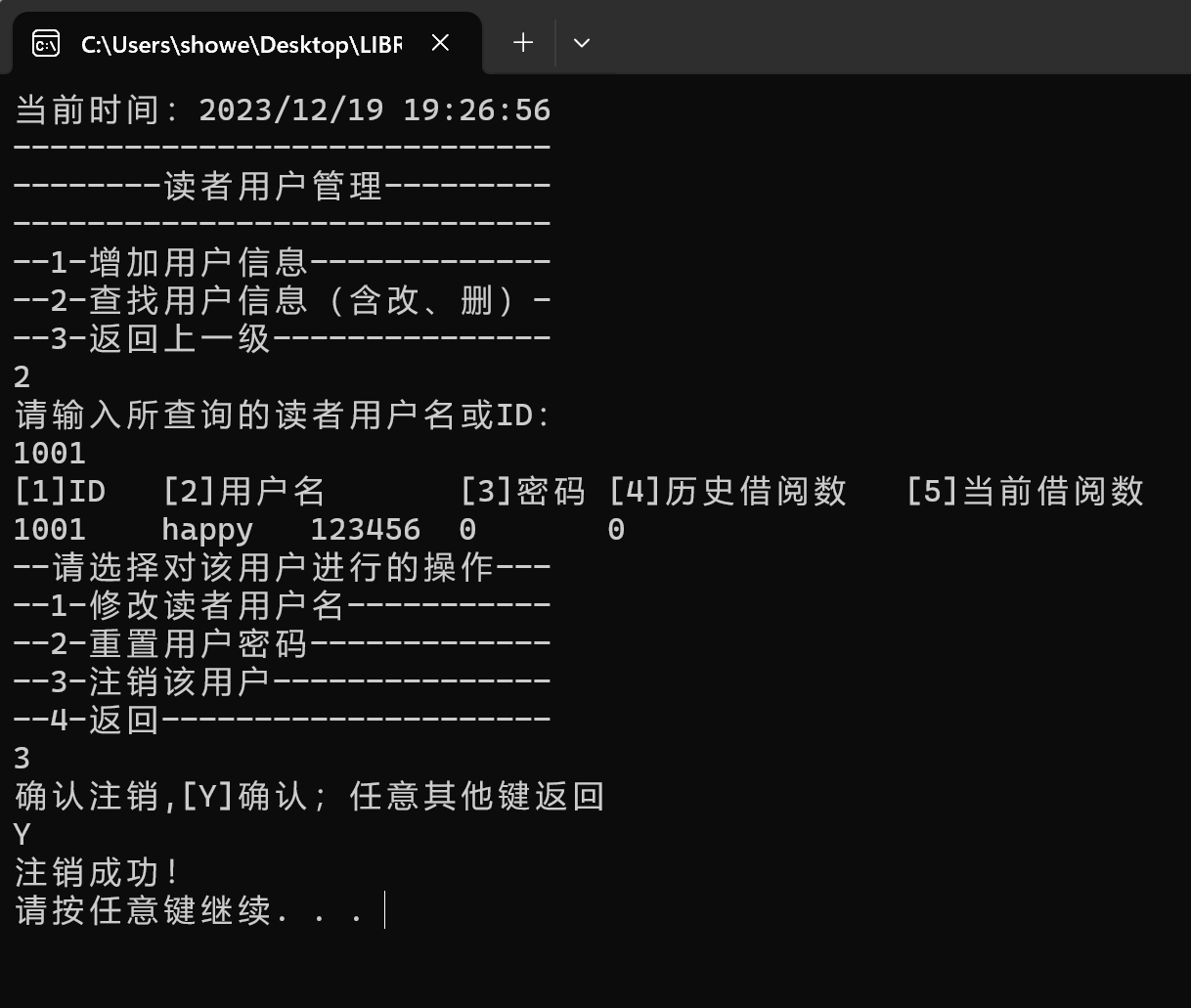
修改用户名



重置密码

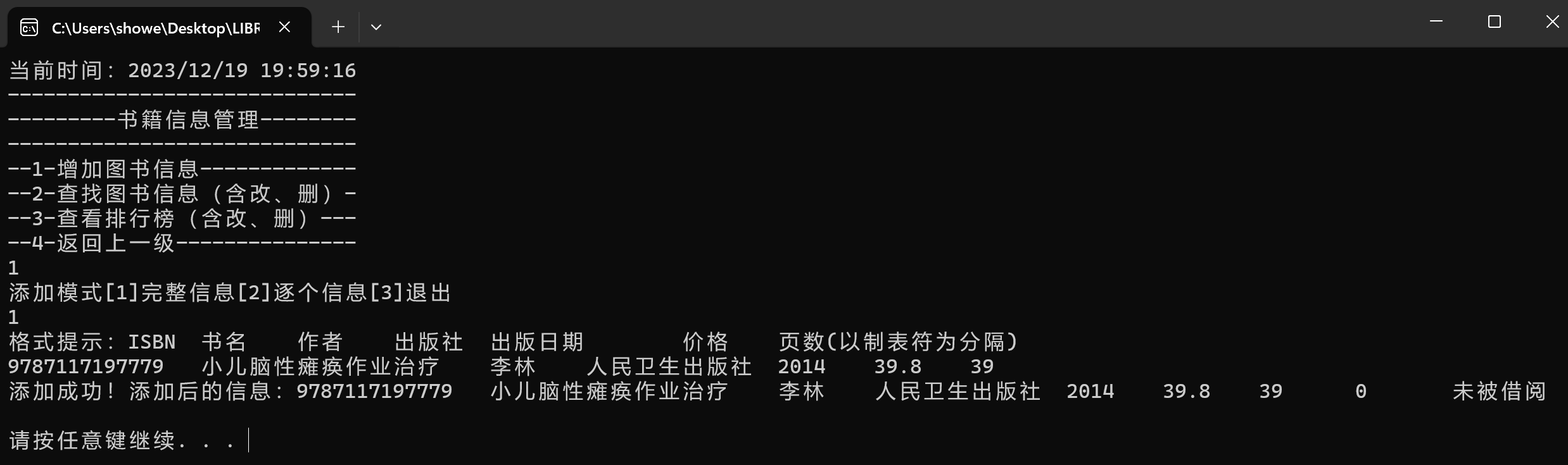


注销账号



**书籍信息管理**

增加图书信息

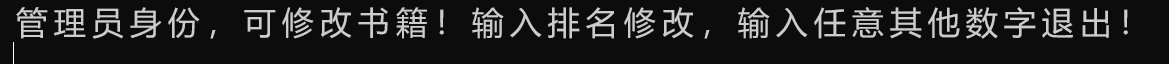


修改图书信息（管理员的搜索界面特殊功能，其他搜索类型均有修改和删除图书功能）



删除图书（管理员的搜索界面特殊功能）

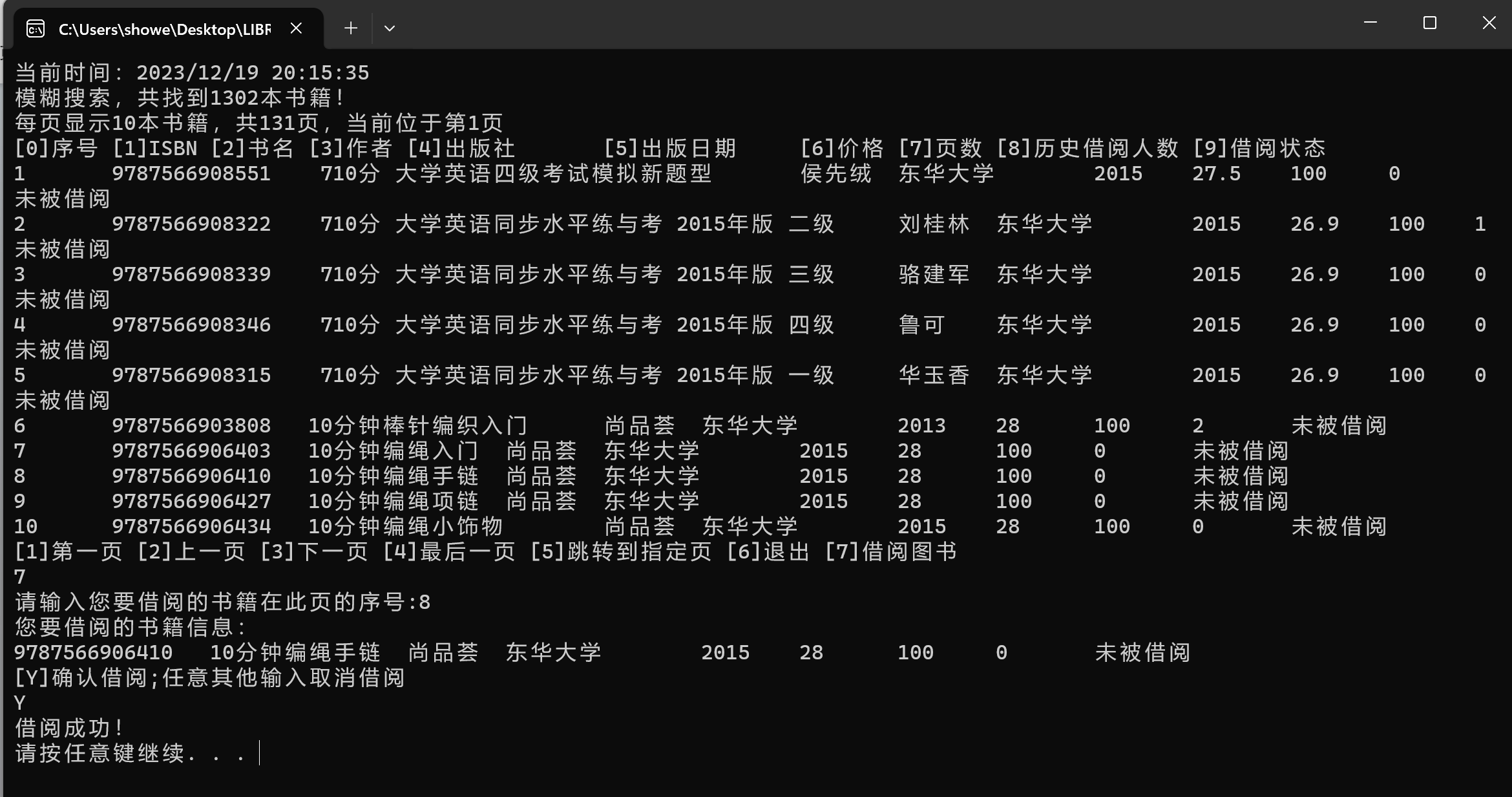


管理员排行榜功能（游客的排行榜功能+可对书籍修改删除，不再详细展示）

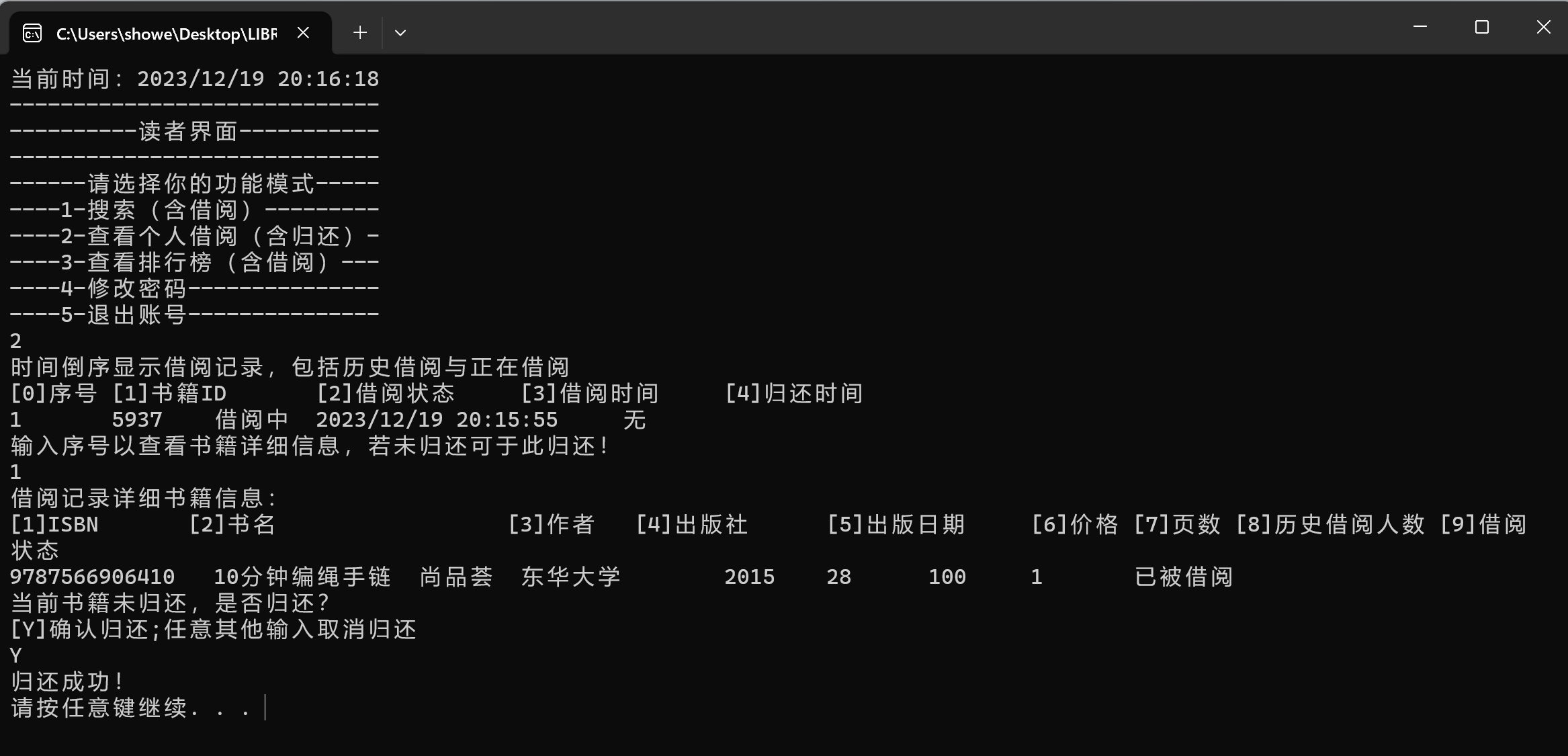
**4.读者模式**

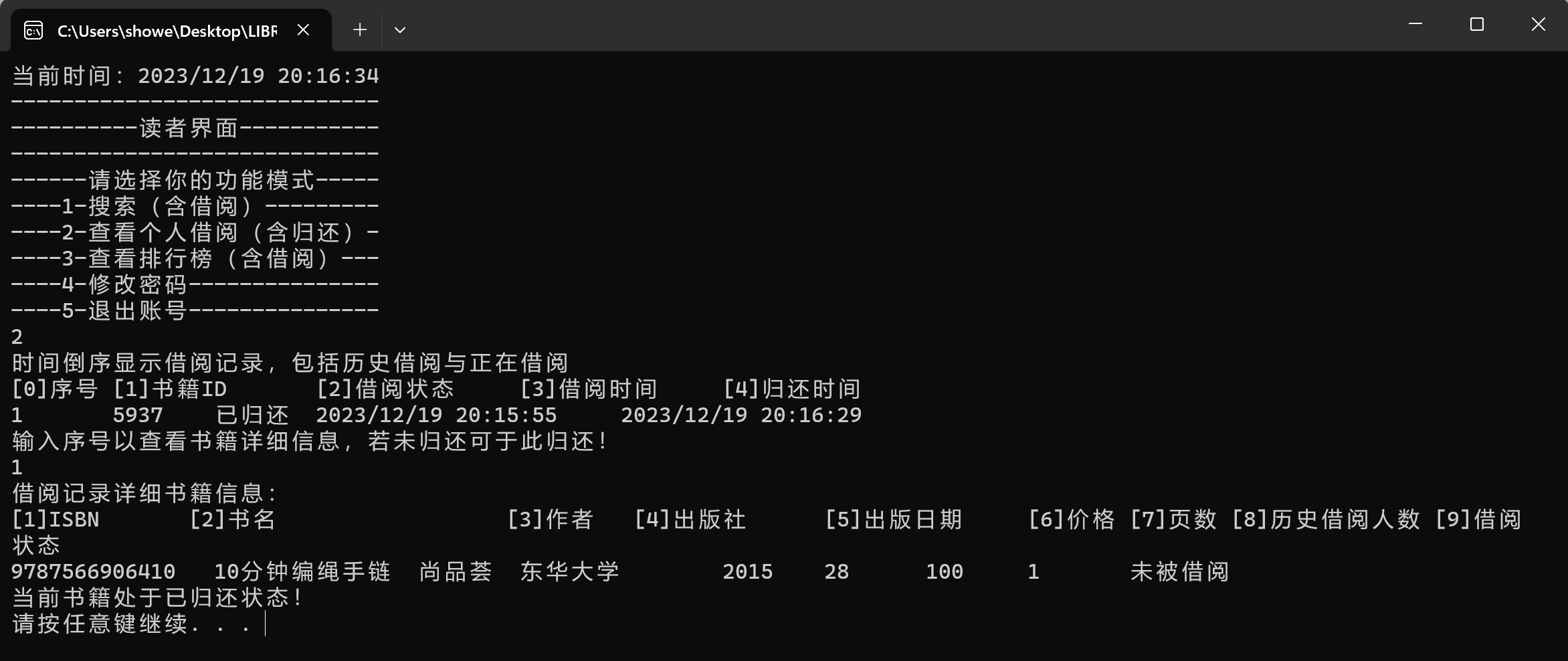
读者注册和读者登录同管理员

**借阅图书**，读者模式搜索特殊功能，其他搜索类型同理

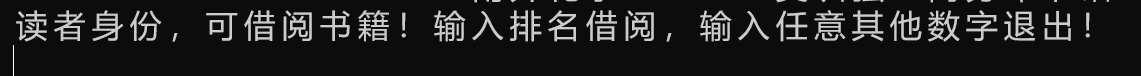


**查看个人借阅记录**，可在此界面归还



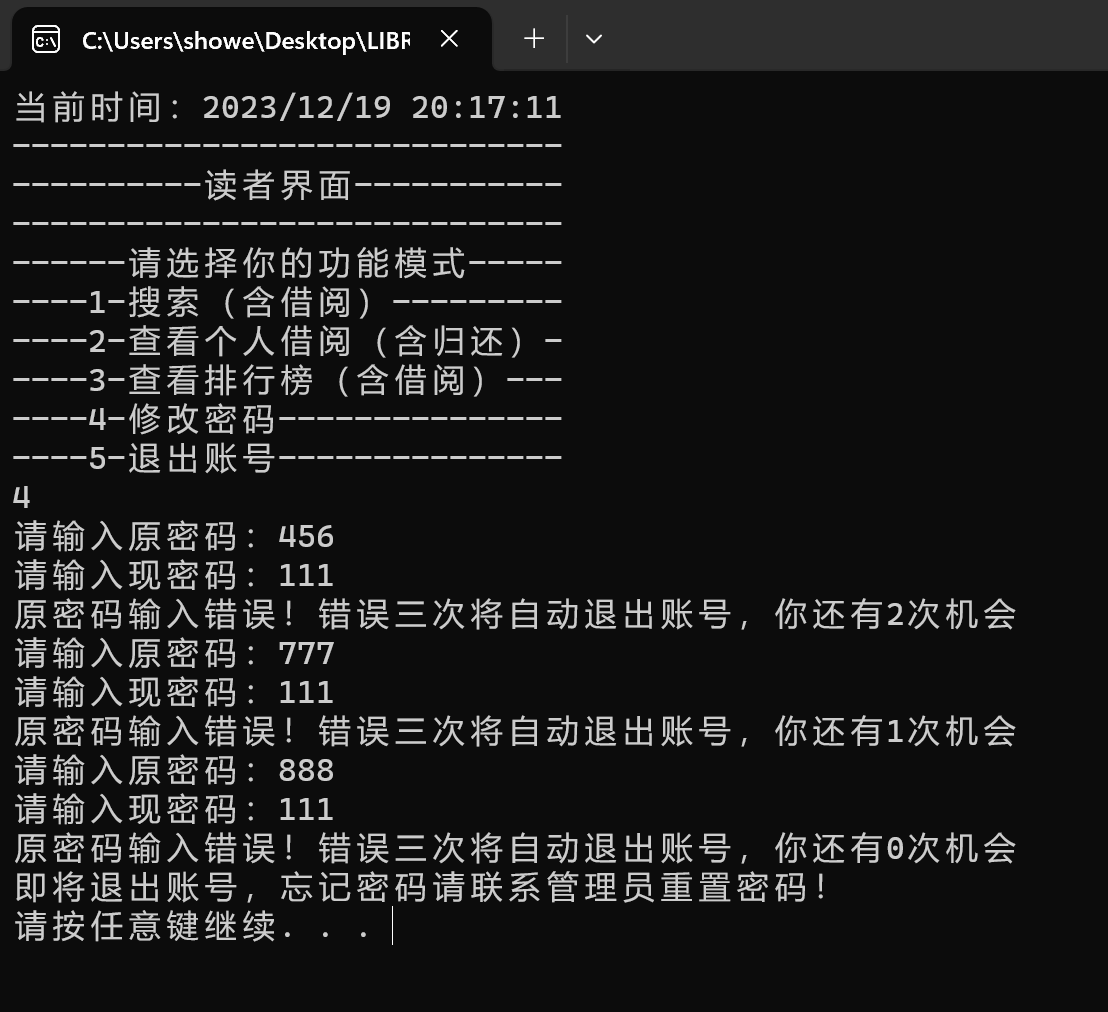


**读者排行榜功能**（游客的排行榜功能+可对书籍借阅，不再详细展示）

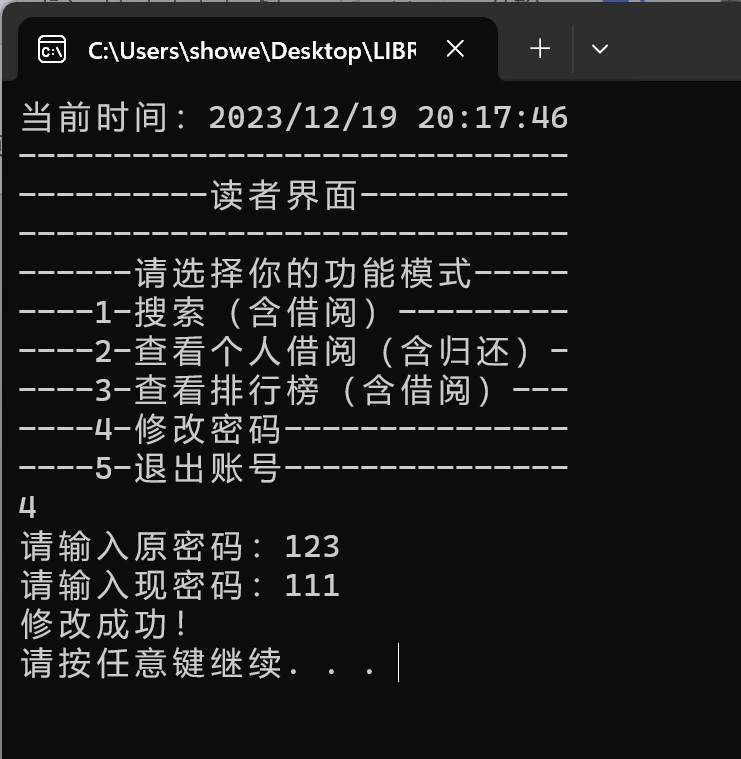


**读者修改密码**

修改错误三次自动退出



修改成功



输入输出展示未包含但已实现的功能：所有界面输入错误的重新输入提示、所有界面的进入上下级功能、大部分功能的操作确认或返回、所有必要的查找和唯一性检验；每一读者同时只可借阅三本书；管理员无法删除借阅中的书籍或读者，保留已删除书籍和读者的借阅记录但无法查看详细信息，保障安全性。

**四、分析和总结**

**1.实验分析**

(1)采用对象组合的方式设计library大类和其他类，main函数创建library对象并调用登录界面作为入口。

(2)实现各功能函数基本全存在library大类，小类仅存书籍和显示函数。

(3)用system(“pause”)对展示的信息停留，system(“cls”)进行清屏，以保证画面美观性。显示当前时间以反映页面刷新。

(4)分析功能之间的层次关系，使得操作符合现实，如借书基于搜书，还书基于查看借阅记录；由于采用的是对象组合的方式，用重载实现多态，使得同一界面不同角色、不同模式有不同的显示和功能

**2.遇到的问题**

(1)类之间的关系和vector的特性。起初想要user、admin类中写各种功能函数，但因为对象组合的形式，无法访问library其他vector容器里的对象，所以只能写在library大类里；整体修改后，仍想要建立user,record,book之间的索引关系，存放彼此的地址，以减少搜索次数，但发现vector删除元素后迭代器会错位，指针会失效。

(2)界面层次设计。输入考虑需要容错性；函数多次调用自身或上一层导致的栈溢出。

(3)读入文件如何以制表符为分隔，如何创造数据文件，以及record文件的格式。

(4)写入文件如何修改特定行的数据。

(5)对类sort排序，要防止频繁修改原有顺序,，起初先将书籍本身全部拷贝，排序后再传递指针，导致借阅后退出函数书籍借阅状态不能更新，重新初始化才能更新。发现bug后怀疑是函数参数的引用问题，研究很久发现实际上是调用函数前的拷贝问题。

(6)添加书籍会乱码

**3.解决的方法**

(1)采取存和传id的方式每次搜索，牺牲一定效率。

(2)while和switch结合，每层都有输入错误的提示，且禁止函数调用自身或调用上一层而采用循环或返回。

(3)运用封装好的split函数分隔，下载网络书籍数据、随机数结合自己写的初始化数据函数创建其他数据，record类储存动态的记录，包含借阅和归还的时间以及当前的状态。

(4)修改特定行采用直接全部覆盖。

(5)采用的是对指针进行排序，再传递指针。

(6)写代码的时候考虑到输入书名等可能含有空格，就使用了getline+cin.ignore，然后导致了乱码，删除后正常；验收时添加了增加同种书籍数量的功能

**4.学习到的新知识**

类与对象除了继承和多态的知识基本都学习、熟悉运用了；熟悉了STL的vector相关操作和面向对象的结合，如push\_back会自动调用拷贝构造函数，find函数第三个参数需要重载运算符，迭代器和指针的关系和区别；分隔函数，获取时间函数的书写；读写文件；界面的设计和优化；问题的分解、提炼、整合等。