四组项目情况展示

注：

1. 请结合目录或大纲查看本文档。

2. 只列出发生变动的内容和新增的内容。

一、补充定义

本周内进行了一些增量开发，下面是相关说明

1. Search类定义

(1). Search类介绍

考虑到对底层树形结构进行直接遍历的时间复杂度较高（约为O(n^2)），使用Search类创建简单的有序索引文件，以方便存取。Search类建立fmindex/fISBNindex+fnameindex+fauthorindex文件，存储每一个数据的关键字（id）及对应的部分值。完成对database类的增删改查后，在**退出系统时**运行。

存储形式：8个字节的ID+ x字节的数据项（fmindex.txt)

6字节的数据项（fISBNindex.txt)

8字节的数据项（fnameindex.txt)

4字节的数据项（fauthorindex.txt)

(2). 定义

```
class Search
{
public:
    //构造函数
    Search();
    //生成索引表，格式为[id 索引项],id为8位整数按二进制字符存储、索引项形式为字符串
```

```
    bool indexFile_Create();
    //indexFile_Create()的变种，生成的索引项为一个字符串（位数截取）
    bool indexFile_Create_mod(string mod);
    //indexFile_Create()的变种，生成的索引项为字符串中自str_min起长度为str_len的部分
    bool indexFile_Create_mod(int str_min, int str_len);
    //调用函数
    bool indexFile_Traverse();
    //调用函数的变种，给定索引项的类型
    bool indexFile_Traverse(string mod);
    //调用函数的变种，给定索引项在字符串中的起始位置与长度
    bool indexFile_Traverse(int str_min, int str_len);
    //遍历函数，对database中所有数据读取到内存
    bool indexFile_Search();
    //初始化函数，确定database中数据关键字的范围（严格单调）
    bool indexFile_Init();
    ~Search();
private:
    int max_key;//最大关键字
    int min_key;//最小关键字
    vector<int>id;//id容器
    vector<string>value;//value字符串容器
};
```

(3). 实现

indexFile_Init()函数：判断树（实际是index文件）中最小的key和最大的key（关键字）

```
bool Search::indexFile_Init() {
    string value;
    while (!database.select(min_key, value)) min_key++;
    while (!database.select(max_key, value)) max_key--;
    return true;
}
```

indexFile_Search()函数：遍历树，将每一个(id,value)分别放入容器id，value。

```
bool Search::indexFile_Search() {
    string v;
    try {
        for (int i = min_key; i < max_key; i++) {
            if (database.select(i, v)) {
                id.push_back(i);
                value.push_back(v);
            }
        }
    }
    catch (std::out_of_range& exc) {
        std::cerr << exc.what() << endl;
        return false;
    }
    return true;
}
```

indexFile_Create()函数：将id与value两个vector容器中的内容逆序存入fintex.txt文件中。

```
bool Search::indexFile_Create() {//全字符串版本
    ofstream findex("findex.txt", ios::binary);
    string data_id;
    string data_value;
    while (!id.empty()) {
        data_id = to_string(id.back());
        id.pop_back();
        data_value = value.back();
        value.pop_back();
        findex.write(data_id.c_str(), LENGTH_ID);
        findex.write(data_value.c_str(), LENGTH_VALUE);
    }
    findex.close();
    return true;
}
```

indexFile_Create_mod(string mod)函数：indexFile_Create()函数的衍生，将id与value两个vector容器中的内容逆序存入f+“mod”+intex.txt文件中，但每个value只截取mod对应的部分，比如ISBN，name，author。注：vector容器读写，遇到空字符会发生std::out_of_range错误。所以需要确保读取到的字符串不为空串（否则会终止生成）。

```
bool Search::indexFile_Create_mod(string mod) {
    string data_id;
    string data_value;
    try {
        if (mod == "ISBN") {
            ofstream fmindex("fISBNindex.txt", ios::binary);
            while (!id.empty()) {
                data_id = to_string(id.back());
                id.pop_back();
                data_value = value.back().substr(START_ISBN, LENGTH_ISBN);
                value.pop_back();
                fmindex.write(data_id.c_str(), LENGTH_ID);
                fmindex.write(data_value.c_str(), LENGTH_ISBN);
                fmindex.close();
            }
        }
        if (mod == "name") {
            ofstream fmindex("fnameindex.txt", ios::binary);
            while (!id.empty()) {
                data_id = to_string(id.back());
                id.pop_back();
                data_value = value.back().substr(START_NAME, LENGTH_NAME);
                value.pop_back();
                fmindex.write(data_id.c_str(), LENGTH_ID);
                fmindex.write(data_value.c_str(), LENGTH_NAME);
                fmindex.close();
            }
        }
        if (mod == "author") {
            ofstream fmindex("fauthorindex.txt", ios::binary);
            while (!id.empty()) {
                data_id = to_string(id.back());
                id.pop_back();
                data_value = value.back().substr(START_AUTHOR, LENGTH_AUTHOR);
                value.pop_back();
```

```
                fmindex.write(data_id.c_str(), LENGTH_ID);
                fmindex.write(data_value.c_str(), LENGTH_AUTHOR);
                fmindex.close();
            }
        }
    }
    catch (std::out_of_range& exc) {//遇到空字符时，读取失败
        std::cerr << exc.what() << endl;
        return false;
    }
    return true;
}
```

indexFile_Create_mod(int str_min, int str_len)函数：indexFile_Create()函数的衍生，将id与value两个vector容器中的内容逆序存入fmintex.txt文件中，同时每个value只截取第str_min位到第str_len位。

```
bool Search::indexFile_Create_mod(int str_min, int str_len) {
    ofstream fmindex("fmindex.txt", ios::binary);
    string data_id;
    string data_value;
    while (!id.empty()) {
        data_id = to_string(id.back());
        id.pop_back();
        data_value = value.back().substr(str_min, str_len);
        value.pop_back();
        fmindex.write(data_id.c_str(), LENGTH_ID);
        fmindex.write(data_value.c_str(), str_len);
    }
    fmindex.close();
    return true;
}
```

indexFile_Traverse()函数：调用上述函数。在main文件中使用。

```
bool Search::indexFile_Traverse() {
    indexFile_Init();
    indexFile_Search();
    indexFile_Create();
    return true;
}
```

```
bool Search::indexFile_Traverse(string mod) {
    indexFile_Init();
    if (indexFile_Search())
        cout << "Search Successfully." << endl;
    else
        cout << "Search Failed" << endl;
    if (indexFile_Create_mod(mod)) {
        cout << "FastIndex create successfully." << endl;
        return true;
    }
    else
    {
        cout << "FastIndex create failed." << endl;
        return false;
```

```
    }
}
```

```
bool Search::indexFile_Traverse(int str_min, int str_len) {
    indexFile_Init();
    if (indexFile_Search())
        cout << "Search Successfully." << endl;
    else
        cout << "Search Failed" << endl;
    if (indexFile_Create_mod(str_min, str_len)) {
        cout << "FastIndex create successfully." << endl;
        return true;
    }
    else
    {
        cout << "FastIndex create failed." << endl;
        return false;
    }
}
```

(4). 调试

```
0. SELECT
1. INSERT
2. REMOVE
3. UPDATE
4. quit
Please input a number:4
Please input the mod:ISBN
Search Successfully.
FastIndex create successfully.
Search traverse successfully.
Bye
```

2. bookdocking类补充定义

对bookdocking类中的bookdocking()、bookadd()、bookdelete()三个函数进行了重载。此部分尚未进行对接与调试。

```
bookdocking::bookdocking(string mod){
    blank_str = string(LENGTH_ID + LENGTH_VALUE, ' ');
    firstEmpty = 0;//firstEmpty shows the first empty destination to store new datas when new data is inputed
    zero_str = "0000";//zero_str is a string to rewrite the address in fEmpty to change the isEmpty information
}


bool bookdocking::bookadd(string mod,int id,string ISBN, string name, string author, string type, string borrowtime, string returntime, string histroy, string onsheelf, string isovertime)
{
    to_id = to_string(id);
    value = ISBN + name + author + type + borrowtime + returntime + histroy + onsheelf + isovertime;
    to_value = value;
    if (database.insert(id, value)) {
        fEmpty.open("fEmpty.txt", ios::binary); //open the file that indicates the empty place in the datafile fIndex
```

```
        if (fEmpty.peek() != EOF) {//check if the fEmpty file is not an empty
one. if so, execute the following codes.
            fData.open("fIndex.txt", ios::binary);//open the file fIndex to add
items
            fEmpty.seekg(0, ios::beg);//go to the head of the fEmpty
            getline(fEmpty, v_index);//read a line (serves as an address (4
bytes data per line), indicates an empty in fIndex
            while (!atoi(v_index.c_str())) {    //find the first non-zero
destination of fEmpty
                firstEmpty = fEmpty.tellg();//store the present destination that
the file pointer fEmpty points, in file fEmpty(read)
                getline(fEmpty, v_index);//id not non-zero, read a new data (in
a new line)
            }
            fData.seekp(atoi(v_index.c_str()), ios::beg);//move the pointer
fData to the empty destination
            //fEmpty.read(c_to_str.c_str(), 4);
            fData.write(to_id.c_str(), LENGTH_ID);//write the id data (within
8(?) bytes) into the fIndex
            fData.write(to_value.c_str(), LENGTH_VALUE);//write the value data
(within 80 bytes) into the fIndex, whithout any blanks between id data
            fData.write("\n", sizeof(char));//start a new line in fIndex for
another new datas
            fEmpty.seekp(firstEmpty, ios::beg);//go to the empty place indicated
in fEmpty
            fEmpty.write(zero_str.c_str(), LENGTH_INDEXADDRESS);//change the
address data to 0000(if trans to int, is 0)
            fData.close();
            fEmpty.close();
            return true;
        }
        else {//if the fEmpty is empty, execute the following codes
            fData.open("fEmpty.txt", ios::binary | ios::app);//no free spaces to
write on, so start a new line at the rear of the file
            fData.write(to_id.c_str(), LENGTH_ID);//write id data
            fData.write(to_value.c_str(), LENGTH_VALUE);//write value data
            //no changes in fEmpty, so no need to change it.
            fData.close();
            fEmpty.close();
            return true;
            //fEmpty.write();
        }
    }
    else
        return false;

}
bool bookdocking::bookdelete(string mod,int id) {
    if (database.remove(id)) {
        fEmpty.open("fEmpty.txt", ios::binary);
        fData.open("fIndex.txt", ios::binary);
        address = search(id);// get the correct destination the fIndex stores
that line of data (return the address of it.)
        v_index = to_string(address);//transform the address(int) to string
        fData.seekp(address, ios::beg);//move the pointer of fData to the place
of address
        fData.write(blank_str.c_str(), LENGTH_ID + LENGTH_VALUE);//replace the
datas with a blank string (88 bytes, a line).
```

```
        if (fEmpty.peek() == EOF) {//if fEmpty is empty
            fEmpty.seekp(0, ios::beg);
            fEmpty.write(v_index.c_str(), LENGTH_INDEXADDRESS);//write the
position data into the fEmpty
            // not [fEmpty.write(new_str.c_str(), LENGTH_INDEXADDRESS);]!!!!
        }
        else {//fEmpty is not empty
            /*
            fEmpty.seekp(0, ios::beg);
            getline(fEmpty, new_str);
            while (atoi(new_str.c_str()) != address) {
                firstEmpty = fEmpty.tellg();
                getline(fEmpty, new_str);
                }
            fEmpty.seekp(firstEmpty, ios::beg);
            fEmpty.write(zero_str.c_str(),LENGTH_INDEXADDRESS);
            */
            fEmpty.seekp(0, ios::beg);
            getline(fEmpty, new_str);
            while (!atoi(new_str.c_str()))//while new_str receives "0000" (0 in
int), keep traversing for the next line
            {
                firstEmpty = fEmpty.tellg();
                getline(fEmpty, new_str);
            }
        }
        return true;
    }
    else
        return false;
}
```

二、进度展示

1. 底层进度

目前的底层部分已经能实现各个功能。底层的数据增删改查已经可以完美运行。而且运行可靠，数据文件和索引文件结构严谨。

(1). 代码实现

索引文件实现展示如下：

```
#include "Cache.h"

Cache::Cache()
{
    cache.resize(CACHESIZE);
    for (int i = 0; i < CACHESIZE; ++i)
        cache[i].valid = false;
}

bool Cache::select(const int key, string &value)
{
    int set = GETSET(key);
    int tag = GETTAG(key);
    if (tag != GETTAG(cache[set].key) || !cache[set].valid)
```

```
        return false;
    value = cache[set].value;
    return true;
}

int Cache::insert(const int key, const string &value, int *oldKey, string
*oldValue)
{
    int set = GETSET(key);
    int tag = GETTAG(key);
    if (tag == GETTAG(cache[set].key) && cache[set].valid)
        return 0;
    int result = 1;
    if (cache[set].valid && cache[set].dirty)
    {
        *oldKey = cache[set].key;
        *oldValue = cache[set].value;
        result = 2;
    }
    cache[set].key = key;
    cache[set].value = value;
    cache[set].valid = true;
    cache[set].dirty = false;
    return result;
}

void Cache::remove(const int key)
{
    int set = GETSET(key);
    int tag = GETTAG(key);
    if (tag == GETTAG(cache[set].key))
        cache[set].valid = 0;
}

bool Cache::update(const int key, string &value)
{
    int set = GETSET(key);
    int tag = GETTAG(key);
    if (tag != GETTAG(cache[set].key) || !cache[set].valid)
        return false;
    cache[set].value = value;
    cache[set].dirty = true;
}

void Cache::save(vector<int> *key, vector<string> *value)
{
    for (int i = 0; i < CACHESIZE; ++i)
    {
        if (cache[i].valid && cache[i].dirty)
        {
            key->push_back(cache[i].key);
            value->push_back(cache[i].value);
        }
    }
}
```

底层实现展示如下：

```
#include "bookdocking.h"

bookdocking::bookdocking(){}



bool bookdocking::bookadd(int id,string ISBN, string name, string author, string
type, string borrowtime, string returntime, string histroy, string onsheelf,
string isovertime)
{
    value = ISBN + name + author + type + borrowtime + returntime + histroy +
onsheelf + isovertime;
    if (database.insert(id, value))
    return true;
    else
    return false;

}
bool bookdocking::bookdelete(int id) {
    if (database.remove(id))
    return true;
    else
    return false;
}

string bookdocking::booksearch(int id) {
    if (database.select(id, value))
    return value;
}

bool bookdocking::bookexist(int id) {
    if (database.select(id, value))
        return true;
    else
        return false;
}
bool bookdocking::bookmodifyISBN(int id, string ISBN) {
    if (database.select(id, value)) {
        value.replace(0, 6, ISBN);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}
bool bookdocking::bookmodifyname(int id, string name) {
    if (database.select(id, value)){
        value.replace(6, 8, name);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
```

```
}
bool bookdocking::bookmodifyauthor(int id, string author) {
    if (database.select(id, value)){
        value.replace(14, 4, author);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}
bool bookdocking::bookmodifytype(int id, string type) {
    if (database.select(id, value)){
        value.replace(18, 4, type);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}
bool bookdocking::bookmodifyborrowtime(int id, string borrowtime) {
    if (database.select(id, value)){
        value.replace(22, 8, borrowtime);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}
bool bookdocking::bookmodifyreturntime(int id, string returntime) {
    if (database.select(id, value)){
        value.replace(30, 8, returntime);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}
bool bookdocking::bookmodifyhistory(int id, string history) {
    if (database.select(id, value)){
        value.replace(38, 40, history);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}
bool bookdocking::bookmodifyonsheelf(int id, string onsheelf) {
    if (database.select(id, value)){
```

```
        value.replace(78, 1, onsheelf);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}bool bookdocking::bookmodifyisovertime(int id, string isovertime) {
    if (database.select(id, value)){
        value.replace(79, 1, isovertime);
        if (database.update(id, value))
            return true;
        else
            return false;
    }
    else
        return false;
}

string dataformatting(int n, string data)
{
    data.append(n - data.size(), ' ');
    return data;
}
```

(2). 调试运行

（仅底层数据部分，**系统整体情况请查看下文逻辑层部分**）

```
1.search
2.add
3.delete
4.modify
0.quit
Please input a number:1
Please input the id:10000001
id: 10000001 ISBN: 010101 name: AAA      author: bbb  type: cccd
borrowtime: 00-00-00
returntime: 00-00-00
histroy:
aaaaaaaaaaaaaa
onsheelf: 1 isovertime: 1
1.search
2.add
3.delete
4.modify
0.quit
Please input a number:0
Bye
```

（以上仅为试运行，底层功能已经实现）

运行后的数据文件内容（二进制形式存储）：

```
   010101AAA     bbb cccd00-00-0000-00-00aaaaaaaaaaaaaa                          11
```

运行后的索引文件内容：

```
 Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: 08 00 00 00 40 00 00 00 00 00 00 00 00 00 00 00    ....@...........
00000010: 01 00 00 00 81 96 98 00 FC FF FF FF                ........|...
```

（有关存储形式的约定，参见之前的s041014与s041022文档）

如上图所示，已经能按预期实现底层功能。

总结，刘云卿在完成索引文件的相关实现与调试、Search类设计、bookdocking类重载的同时完成了对底层的调试。全子修完成了对数据部分的实现与调试，并帮助逻辑层小组同学编写函数、进行调试，成功解决逻辑层技术上的难题，使得图书系统可以运行。

2. 逻辑层进度

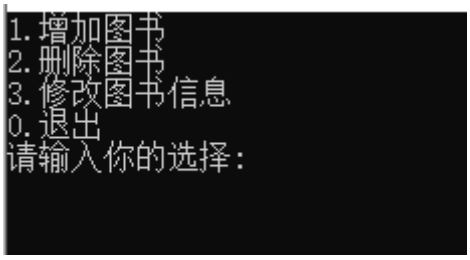
付可意和胡新月同学实现了管理员系统的命令行版本以备用。其中完成了对用户的增删改查和对图书的增删改函数，经过测试，命令行版本已经实现。计划着手与界面UI层的对接。

孙久杰同学基本实现了借阅系统的命令行版本。对其中用户登陆、注册功能，用户根据ID查书、借书功能，用户查看、修改个人信息功能，用户查看已借图书功能完成初步的测试。计划与UI层对接。

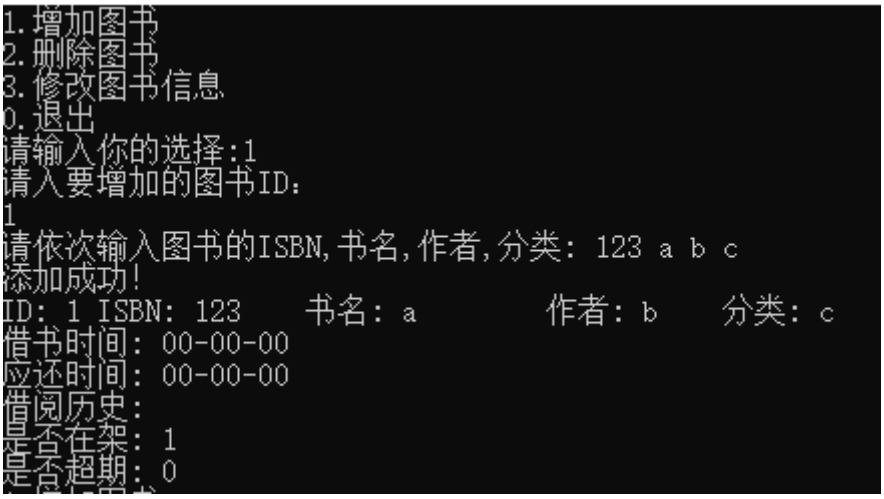
逻辑层的实现在底层的基础上。

运行截图：**（目前为控制台）**

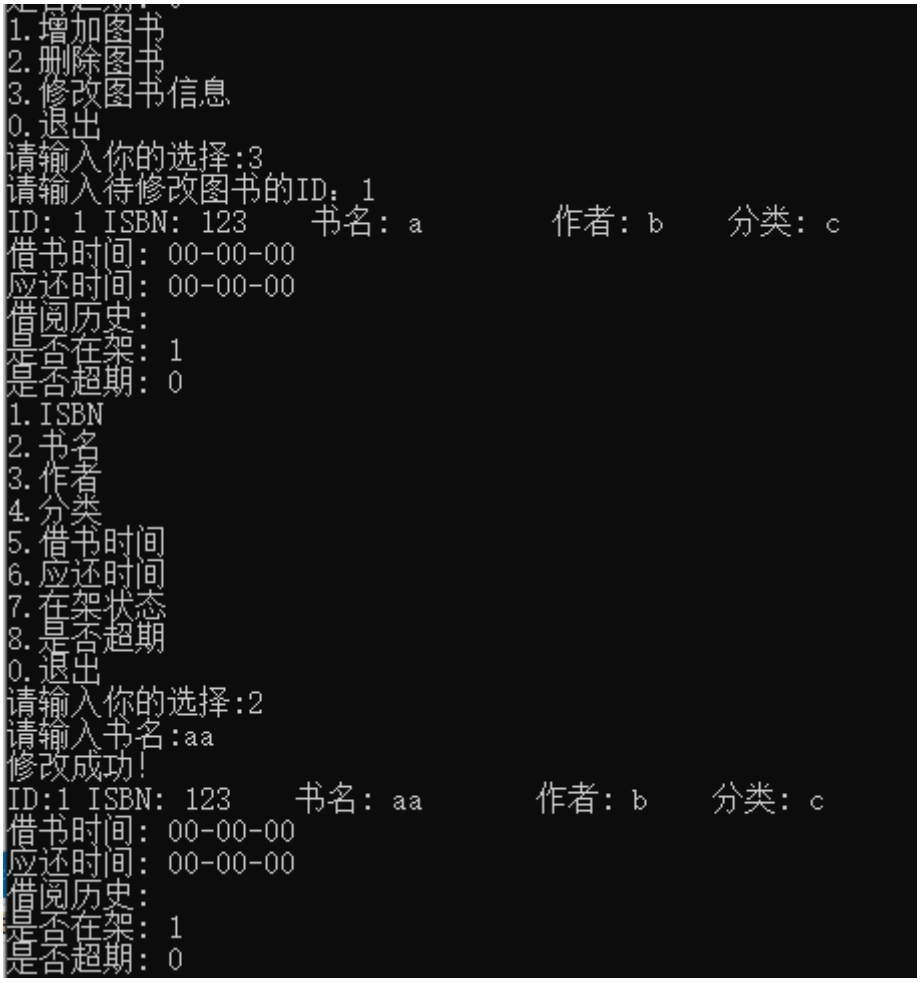
图书管理界面



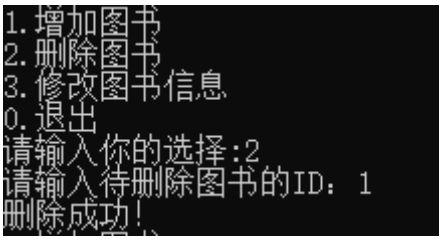
增加图书



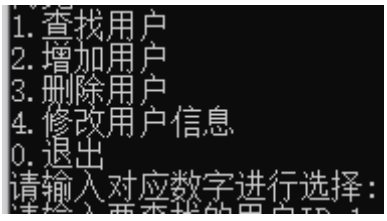
修改图书



删除图书



用户管理菜单界面



增加用户

```
1.查找用户
2.增加用户
3.删除用户
4.修改用户信息
0.退出
请输入对应数字进行选择:2
请输入要增加的用户ID:1
请输入要增加的用户姓名 大学 专业 密码 邮箱:a ouc cs 12345 123@qq.com
增加用户成功!
该用户当前信息如下:
1.ID: 1
2.姓名: a
3.学校: ouc
4.专业: cs
5.密码: 12345
6.邮箱: 123@qq.com
```

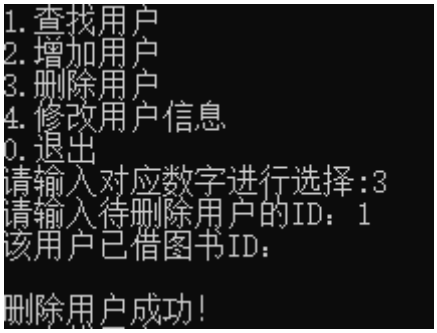
查找用户

```
1.查找用户
2.增加用户
3.删除用户
4.修改用户信息
0.退出
请输入对应数字进行选择:1
请输入要查找的用户ID:1
该用户当前信息如下:
1.姓名:1
2.学校:a
3.专业:b
4.密码:c
5.邮箱:d
```

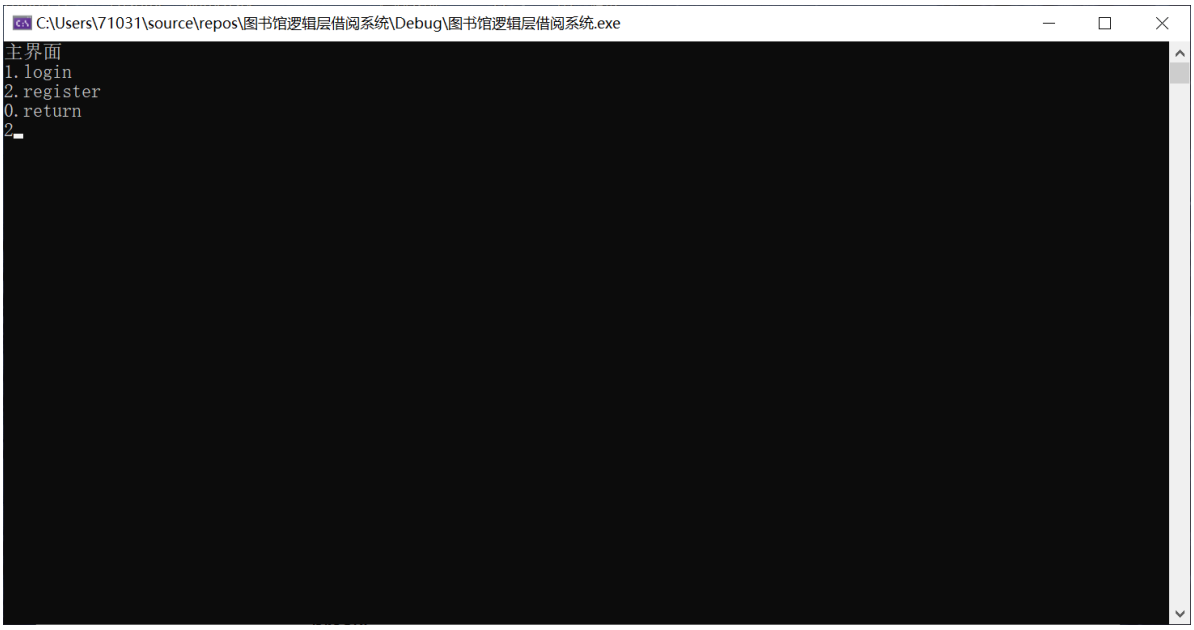
修改用户

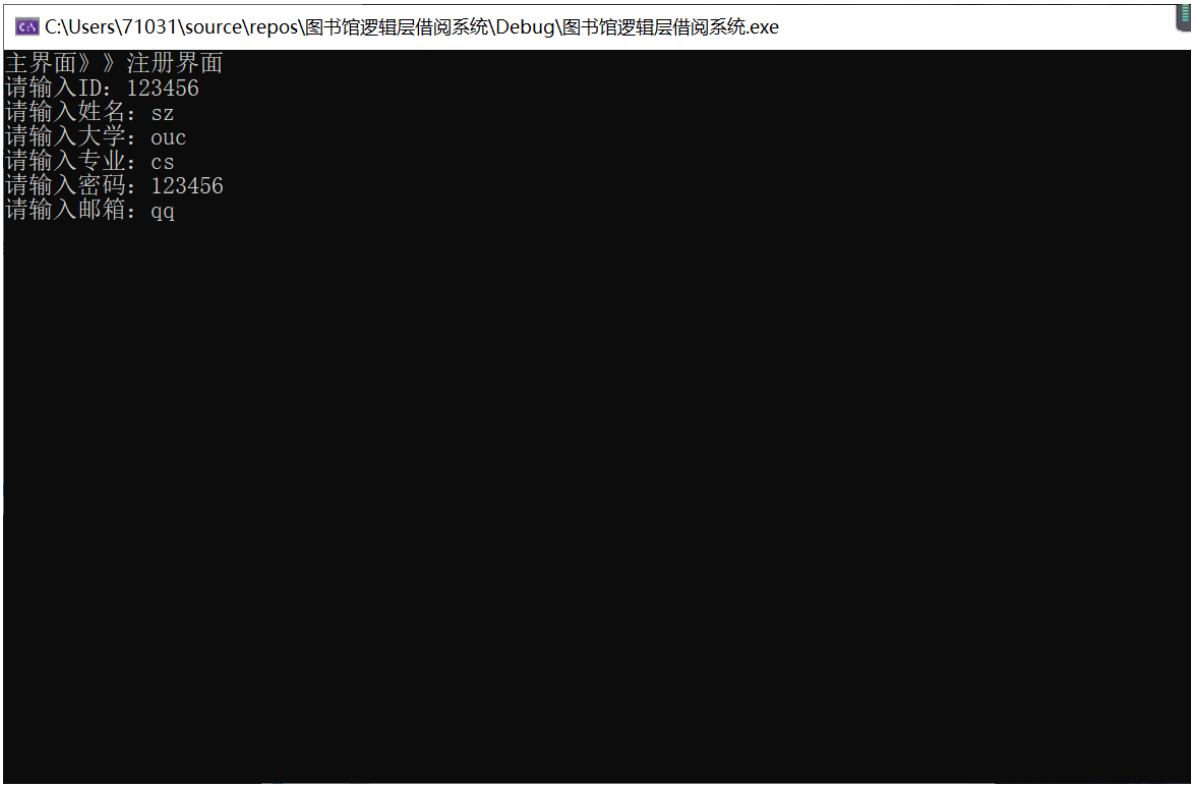
```
1.查找用户
2.增加用户
3.删除用户
4.修改用户信息
0.退出
请输入对应数字进行选择:4
请输入要修改用户的ID: 1
该用户当前信息如下:
1.姓名:a
2.学校:ouc
3.专业:cs
4.密码:12345
5.邮箱:123@qq.com
可修改的用户信息:
1.姓名
2.学校
3.专业
4.密码
5.邮箱
0.退出
请输入对应数字进行修改:3
请依次输入要修改用户的专业:gis
用户信息修改成功!
该用户当前信息如下:
1.姓名:a
2.学校:ouc
3.专业:gis
4.密码:12345
5.邮箱:123@qq.com
```

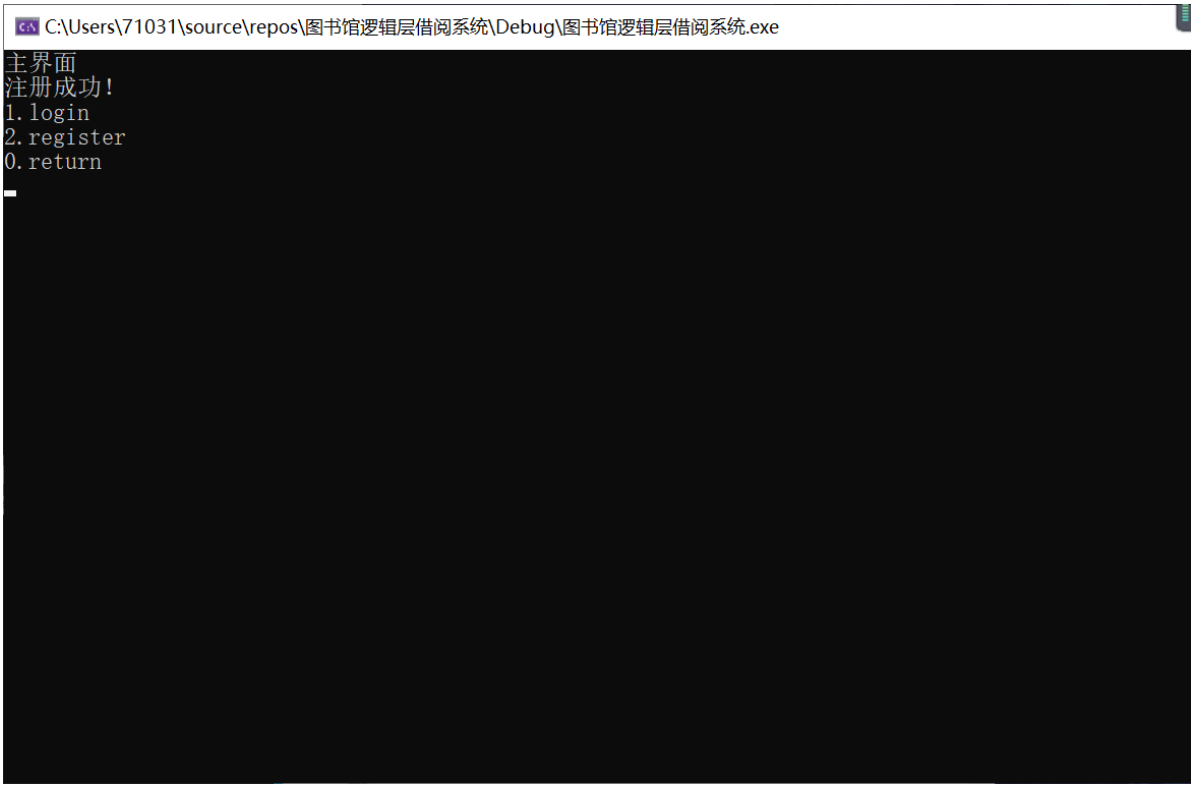
删除用户


1.查找用户
2.增加用户
3.删除用户
4.修改用户信息
0.退出
请输入对应数字进行选择:3
请输入待删除用户的ID：1
该用户已借图书ID：
删除用户成功!

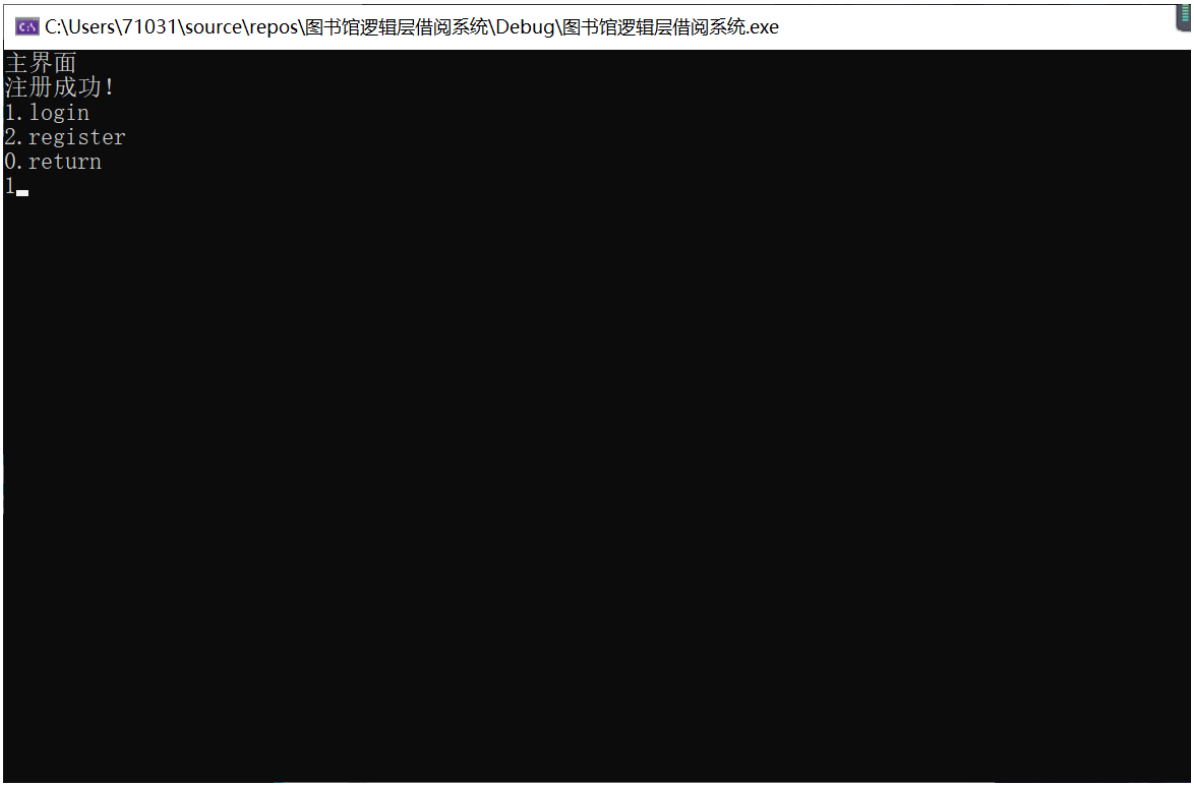
注册

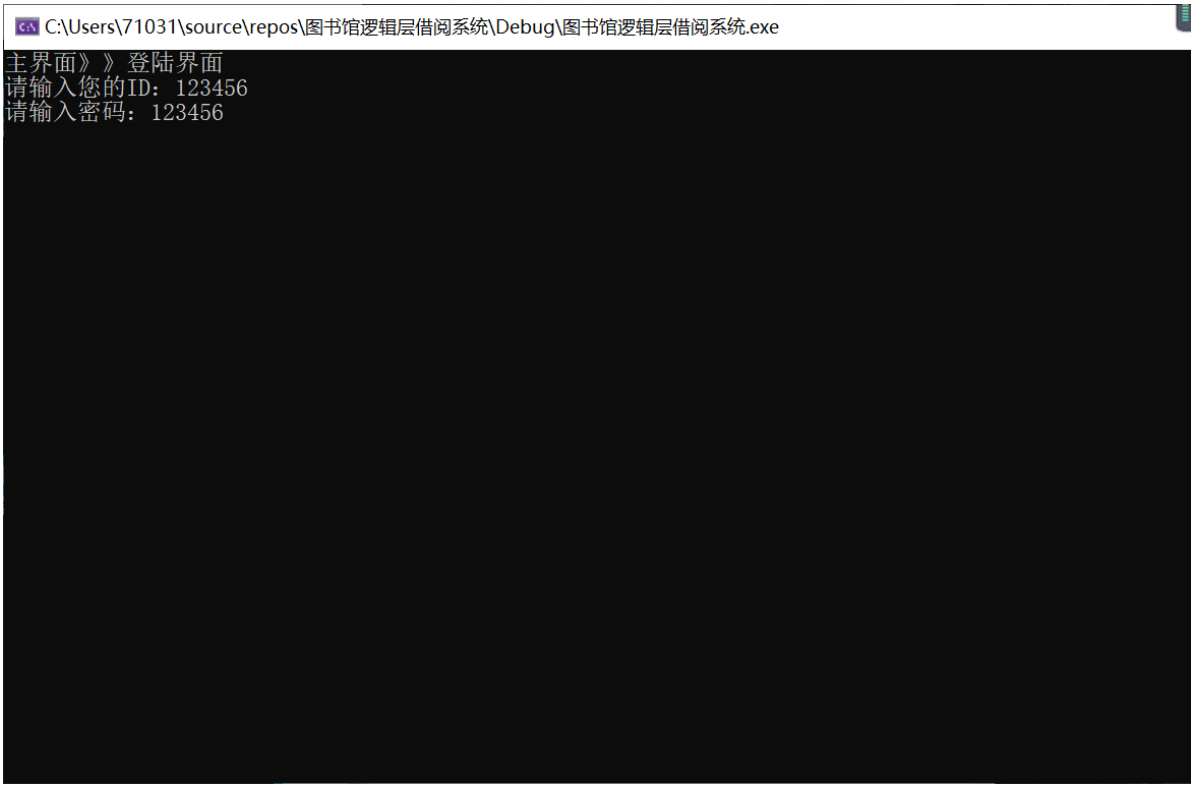

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面
1.login
2.register
0.return
2

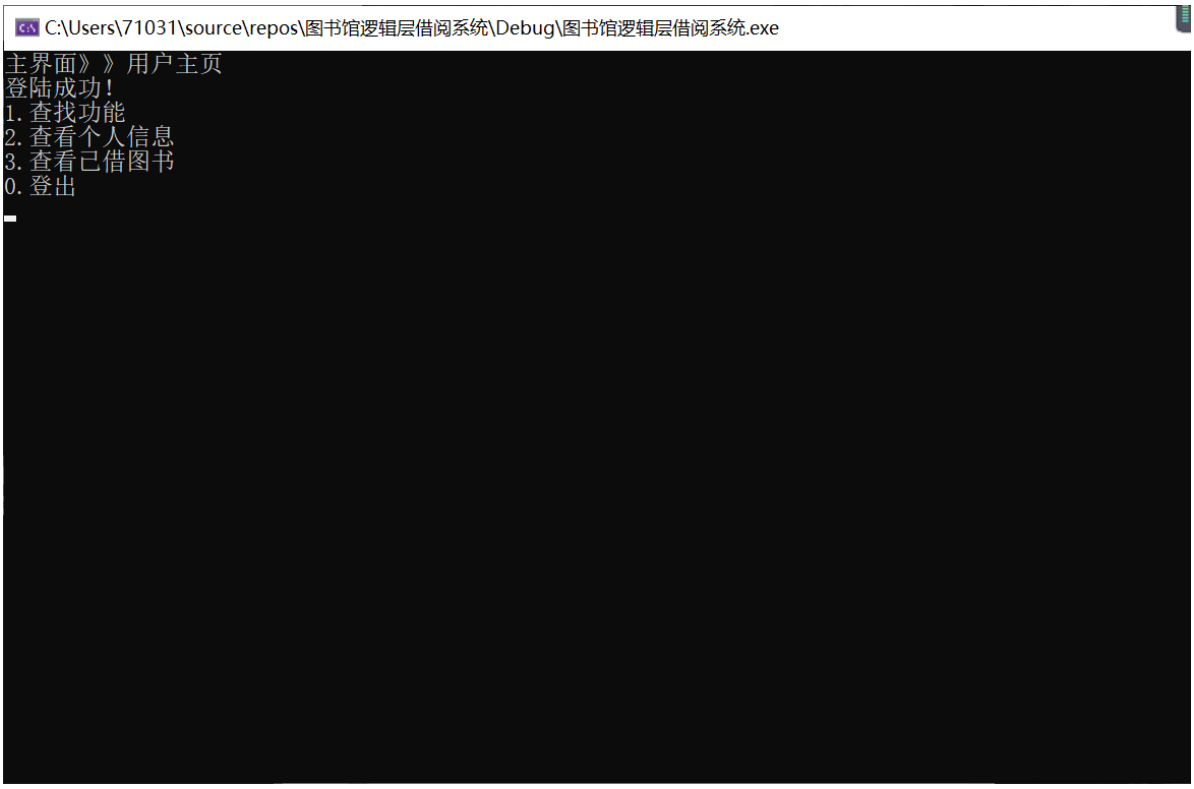

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》注册界面
请输入ID：123456
请输入姓名：sz
请输入大学：ouc
请输入专业：cs
请输入密码：123456
请输入邮箱：qq


C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面
注册成功!
1.login
2.register
0.return

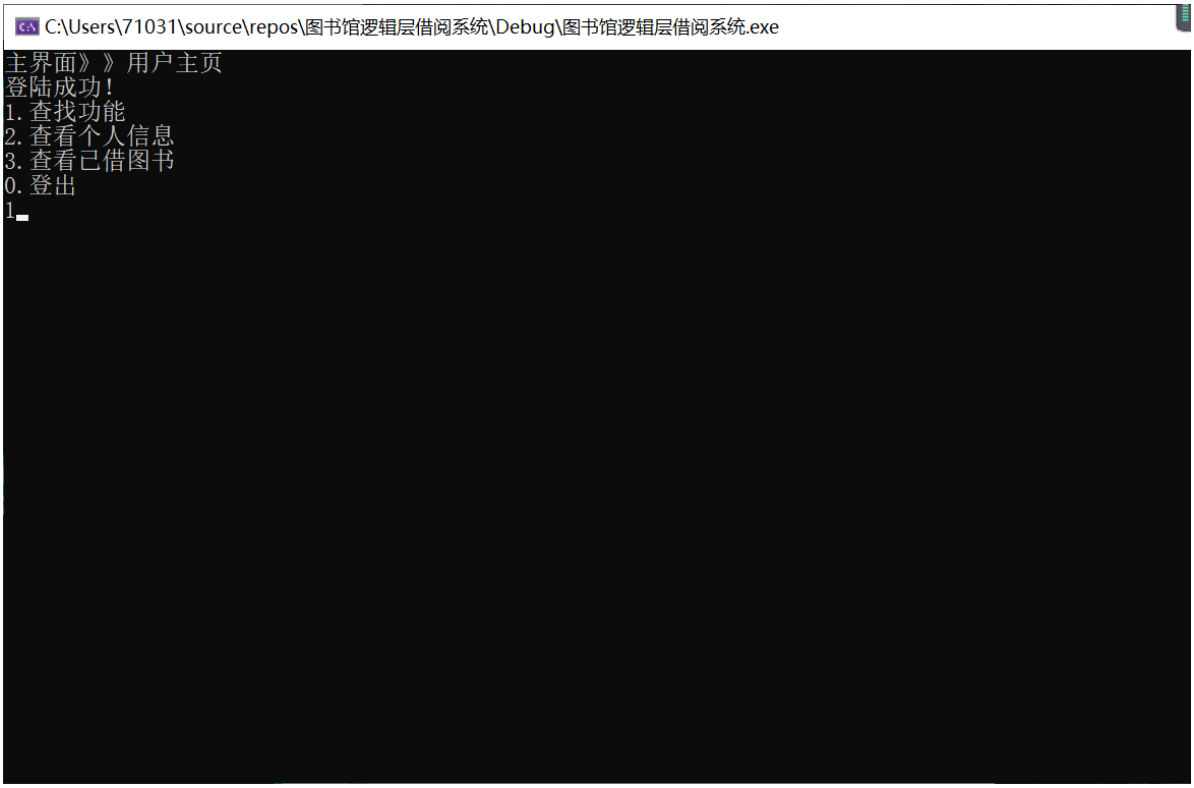
登录

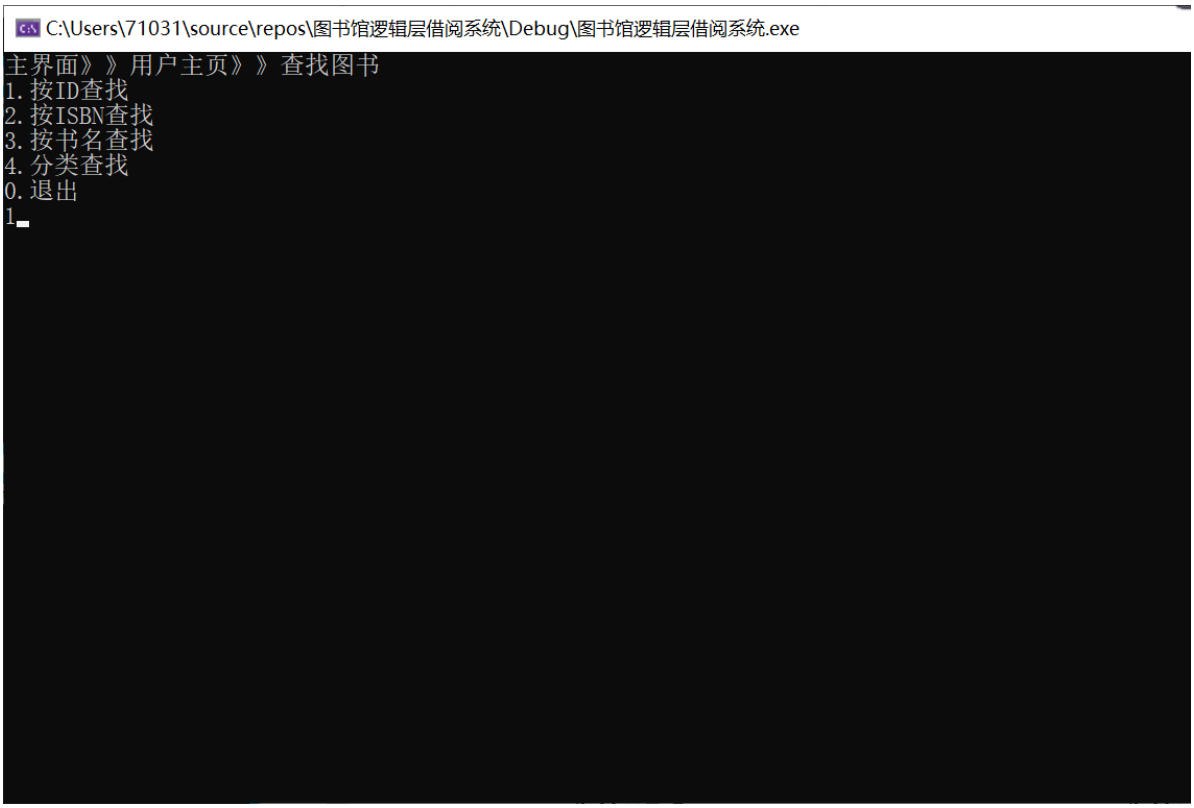

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面
注册成功!
1.login
2.register
0.return
1

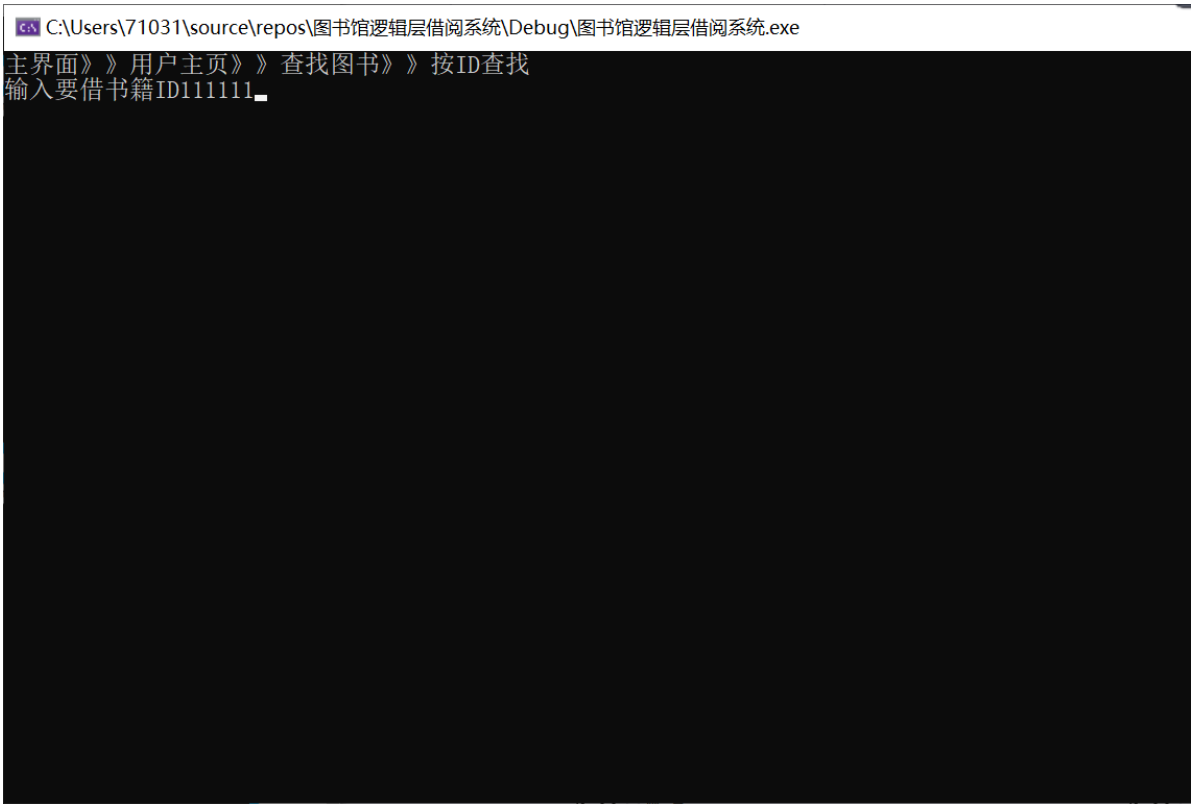

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》登陆界面
请输入您的ID：123456
请输入密码：123456

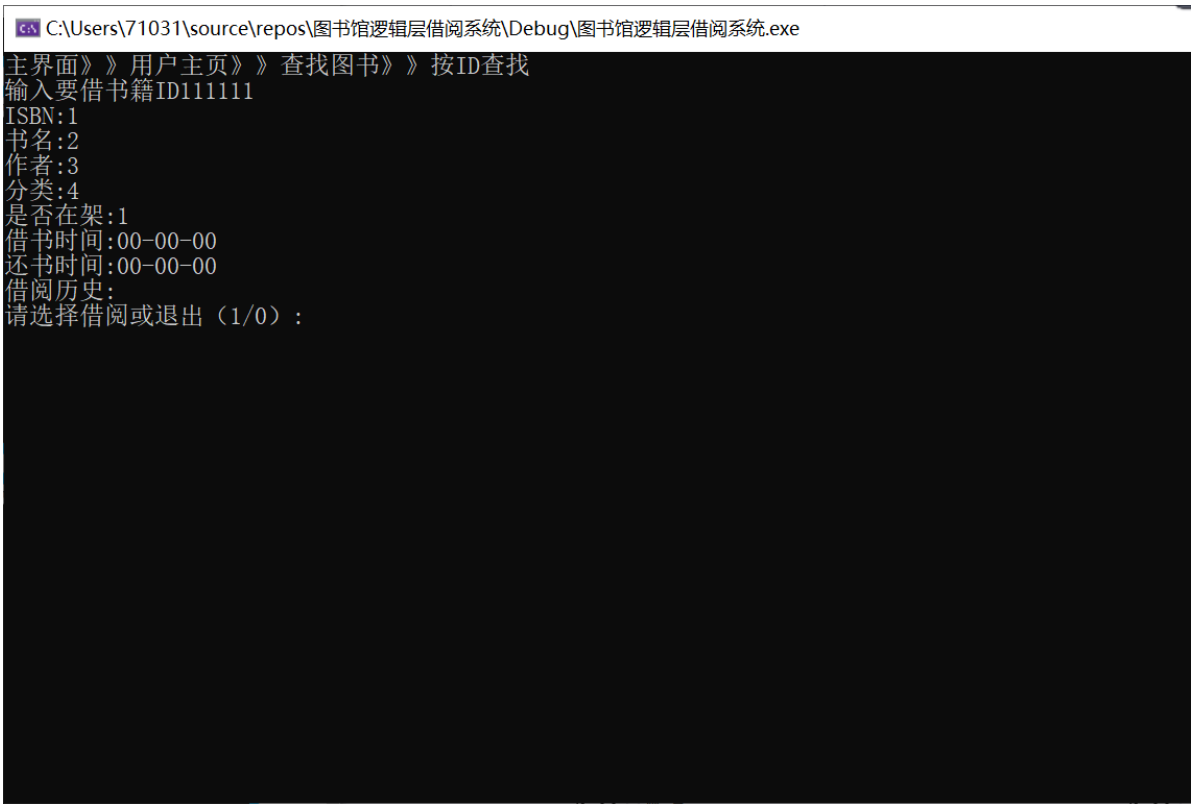

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页
登陆成功！
1. 查找功能
2. 查看个人信息
3. 查看已借图书
0. 登出

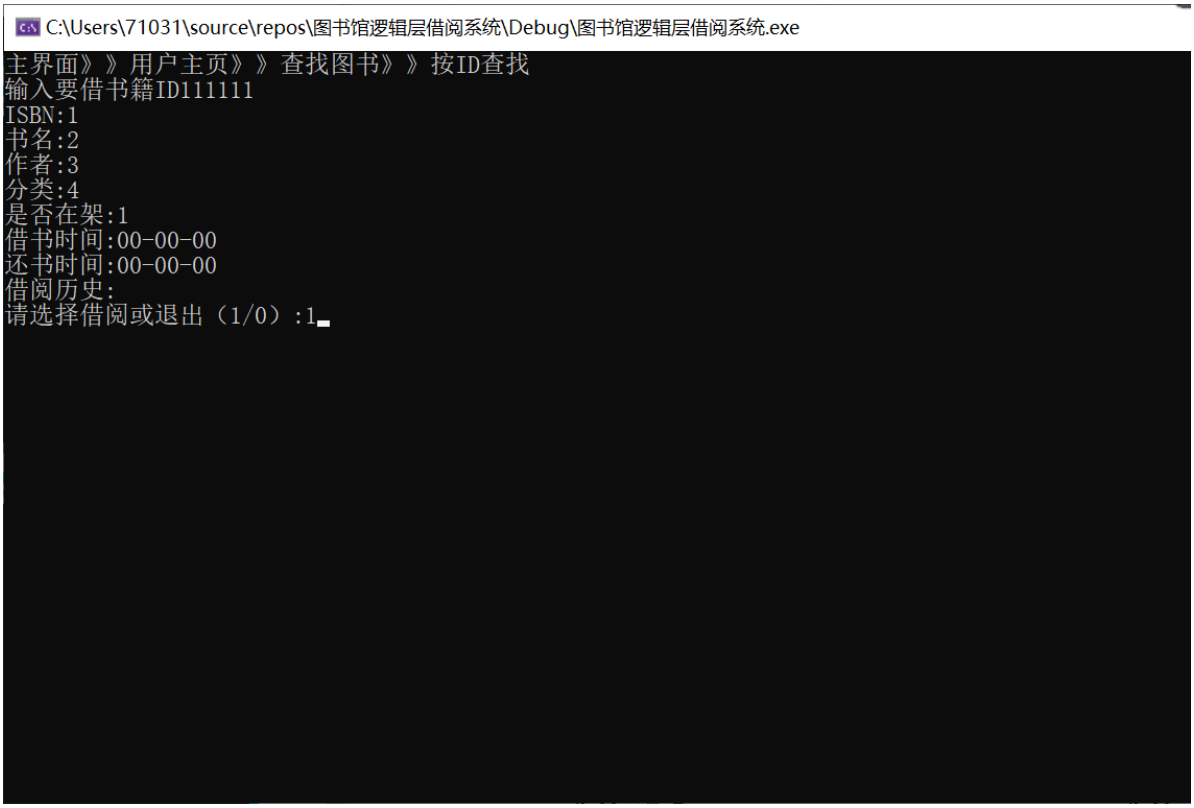
查书

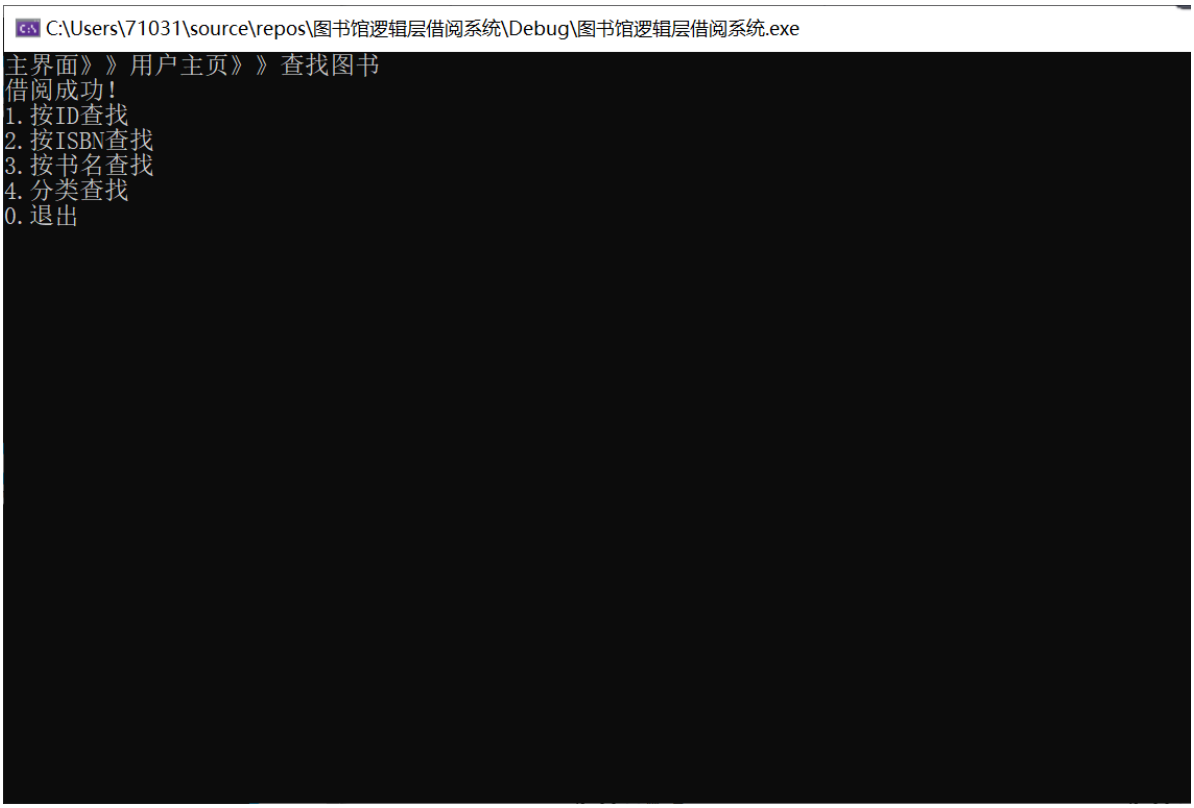

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页
登陆成功!
1.查找功能
2.查看个人信息
3.查看已借图书
0.登出
1


C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页》》查找图书
1.按ID查找
2.按ISBN查找
3.按书名查找
4.分类查找
0.退出
1

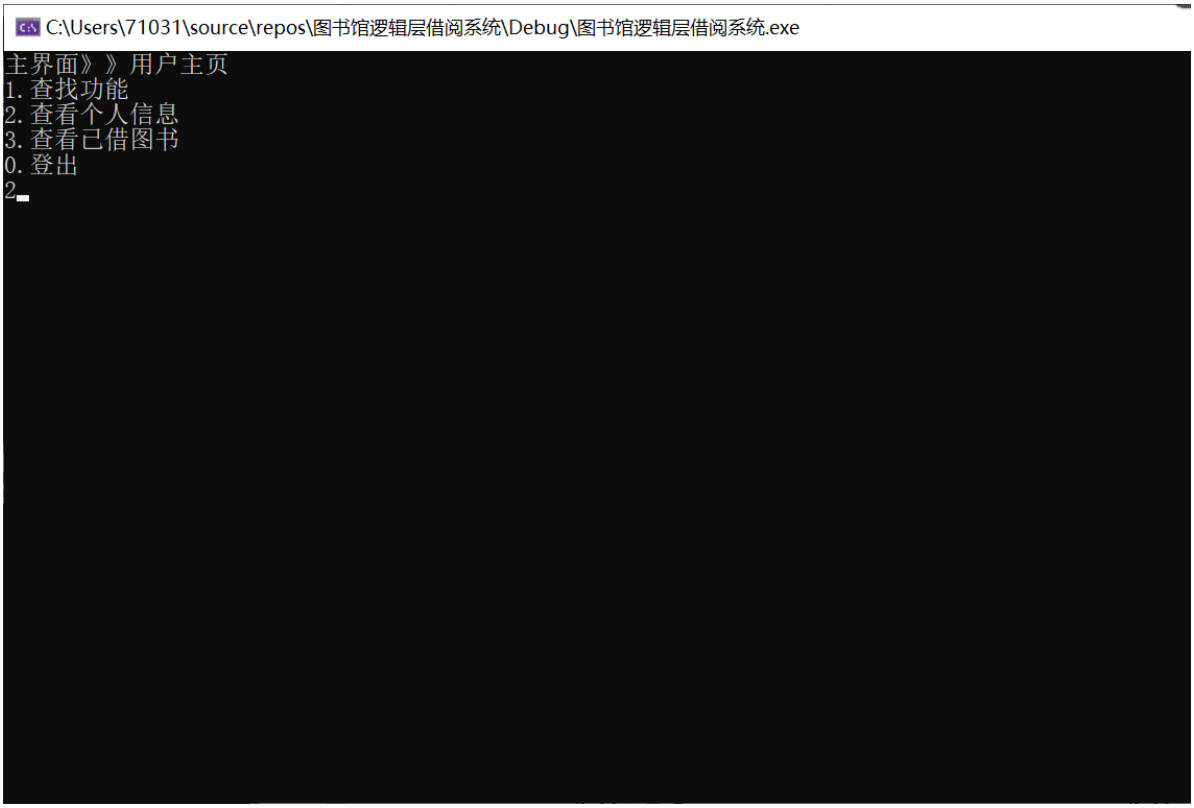

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页》》查找图书》》按ID查找
输入要借书籍ID111111


C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页》》查找图书》》按ID查找
输入要借书籍ID111111
ISBN:1
书名:2
作者:3
分类:4
是否在架:1
借书时间:00-00-00
还书时间:00-00-00
借阅历史:
请选择借阅或退出（1/0）：


C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页》》查找图书》》按ID查找
输入要借书籍ID111111
ISBN:1
书名:2
作者:3
分类:4
是否在架:1
借书时间:00-00-00
还书时间:00-00-00
借阅历史:
请选择借阅或退出（1/0）:1

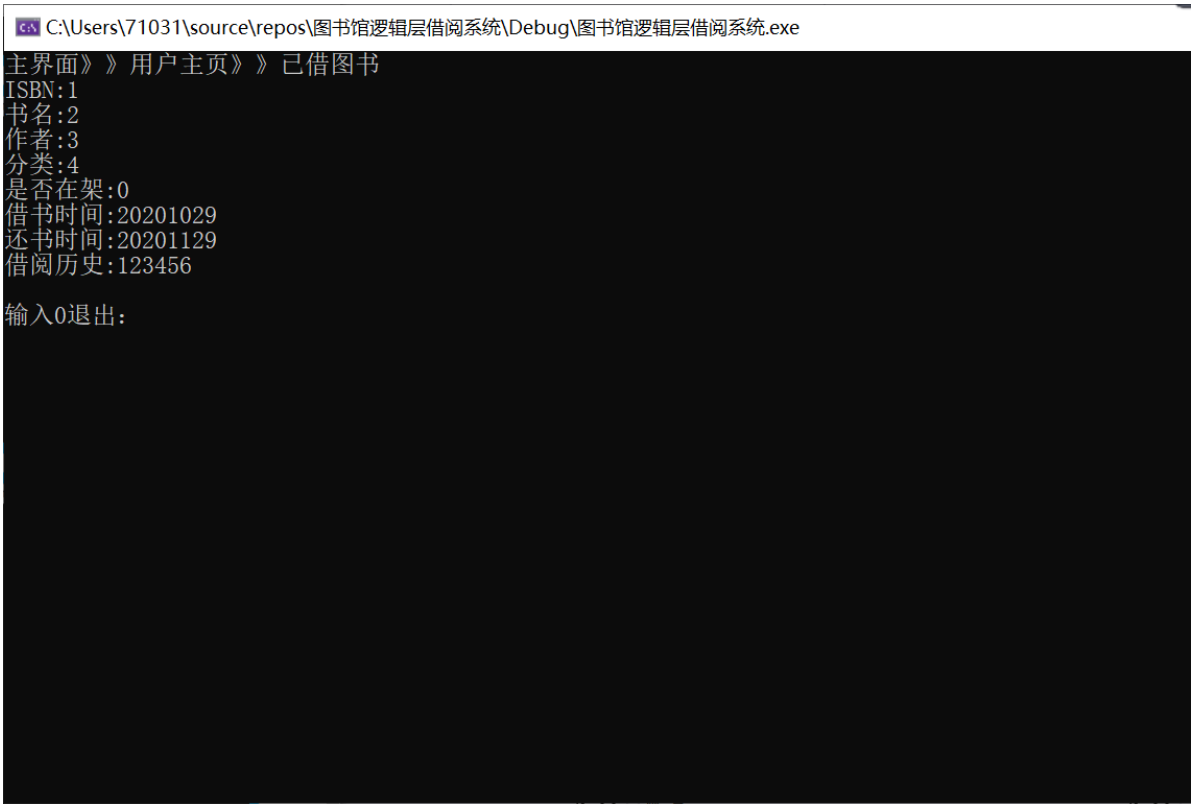

C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页》》查找图书
借阅成功!
1.按ID查找
2.按ISBN查找
3.按书名查找
4.分类查找
0.退出

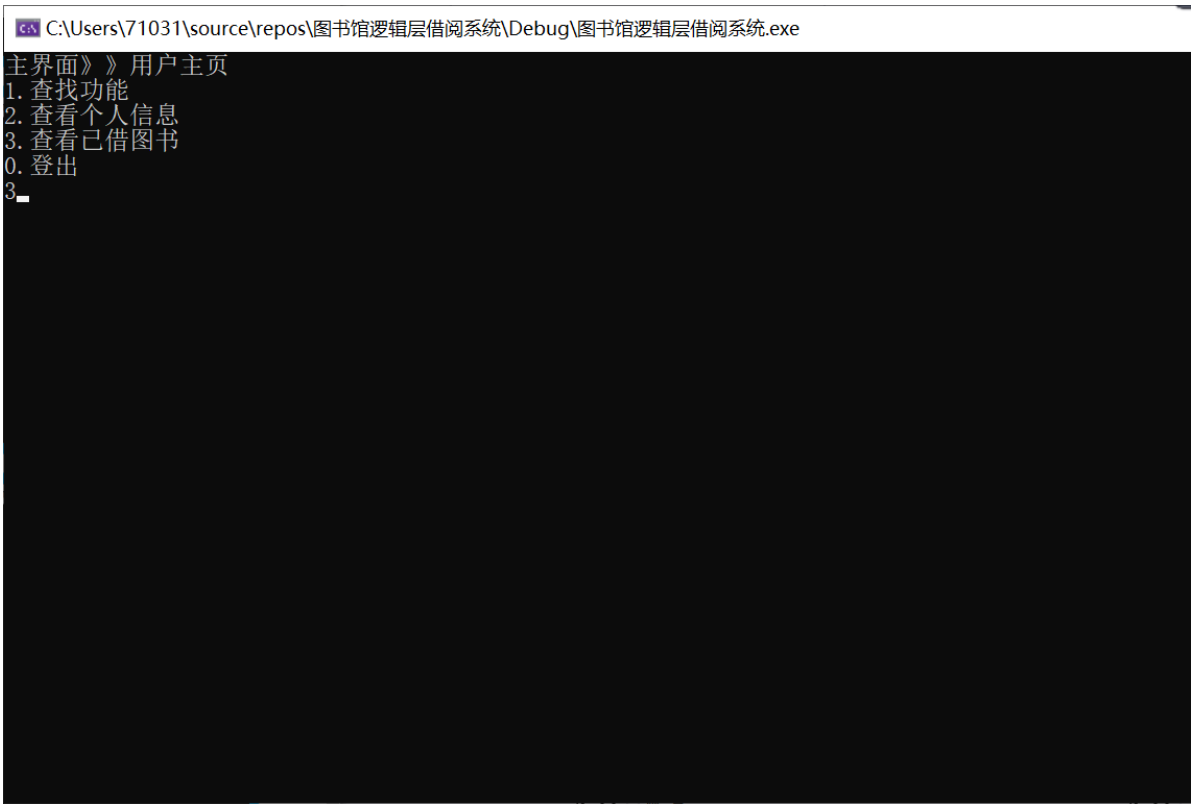
个人信息


C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页
1.查找功能
2.查看个人信息
3.查看已借图书
0.登出
2


C:\Users\71031\source\repos\图书馆逻辑层借阅系统\Debug\图书馆逻辑层借阅系统.exe
主界面》》用户主页》》个人信息
0.用户ID：123456
1.用户名字：sz
2.用户大学：ouc
3.用户专业：cs
4.用户邮箱：qq
修改信息（修改密码请输入5,-1返回上级）：

已借书籍查询

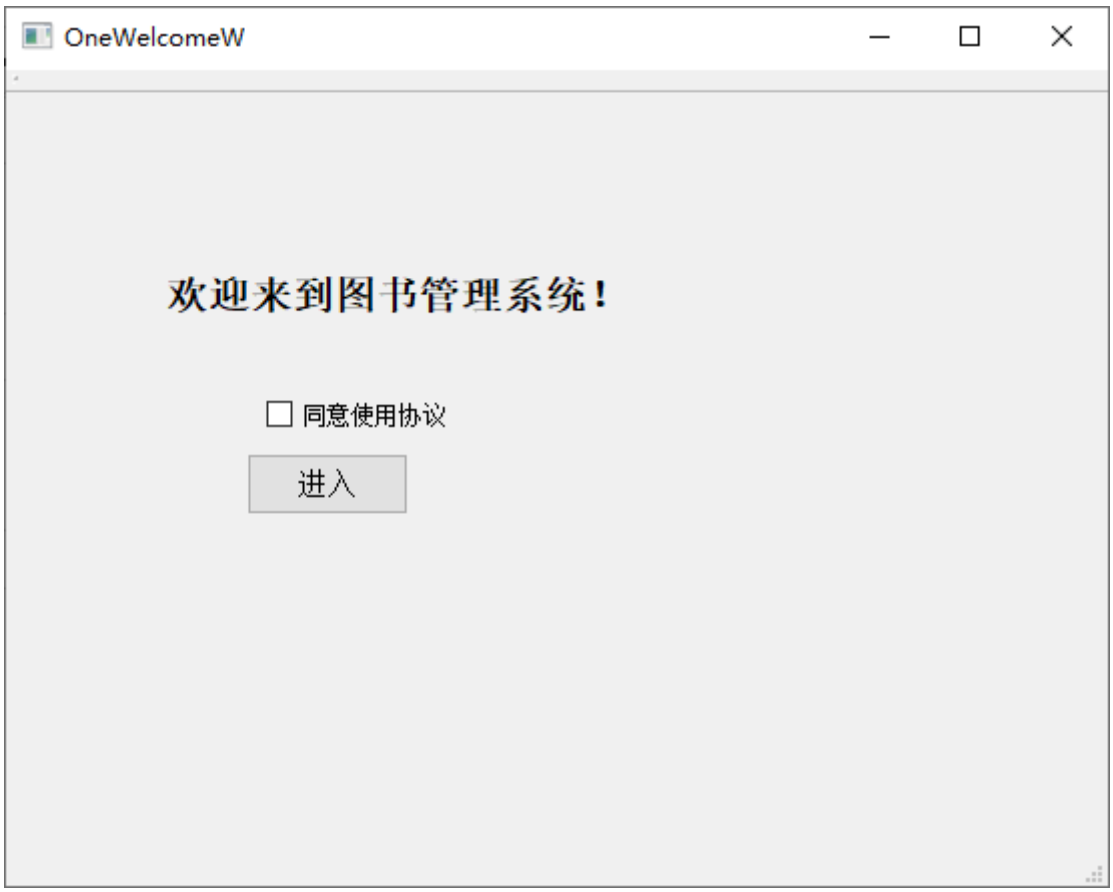




3. UI层进度

毛鸿麟与寇金娣在本周内将所有界面根据已经实现的底层和逻辑进度全部设计完毕，并且界面整合完毕，但还没实现美化界面。和逻辑层的接口已经和逻辑层交流并完成文档说明。接下来是整个项目的移植以及Qt界面在代码上实现和逻辑层的对接。

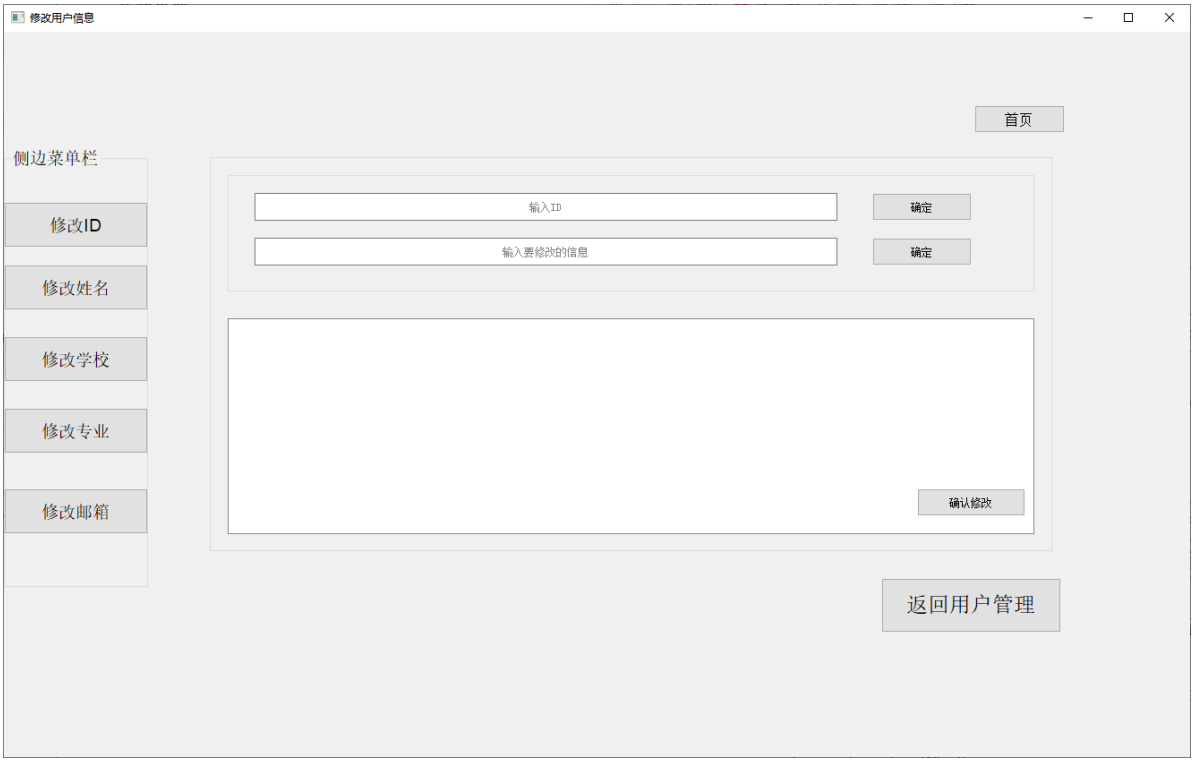
(1).运行截图

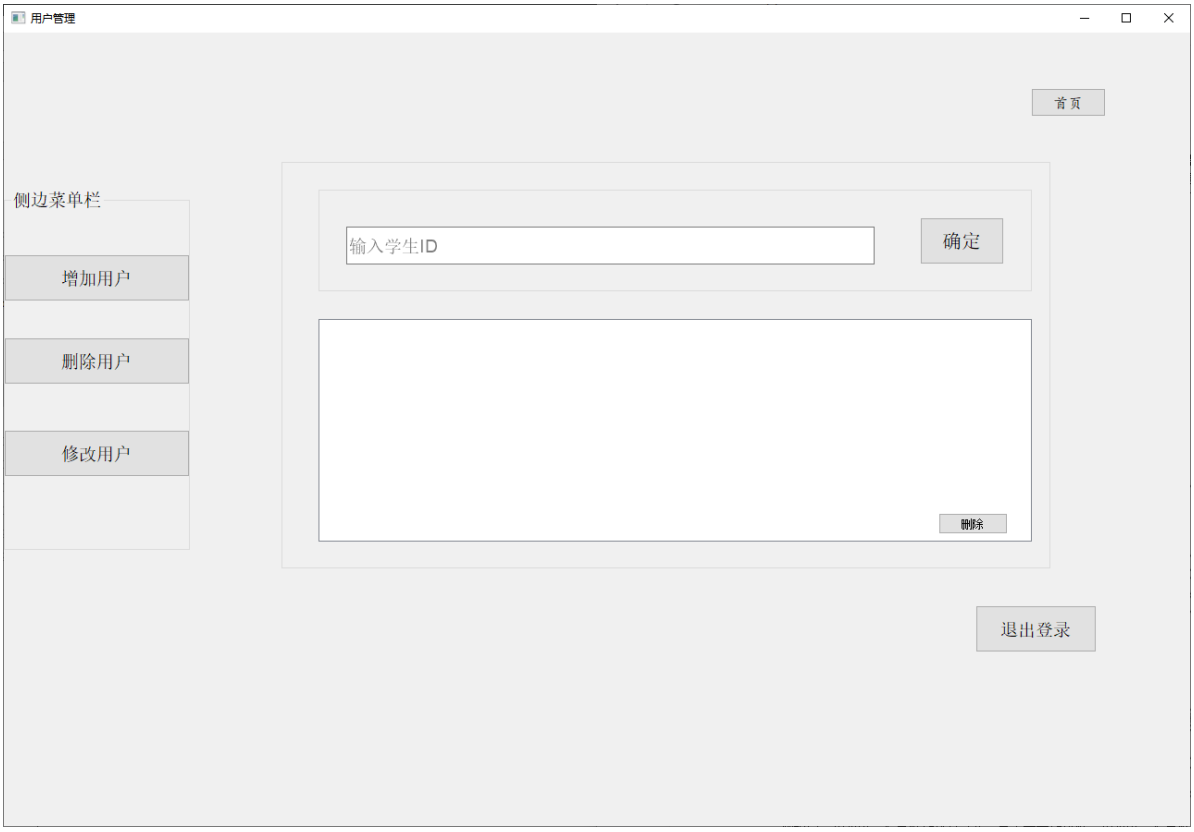

OneWelcomeW
欢迎来到图书管理系统！
同意使用协议
进入

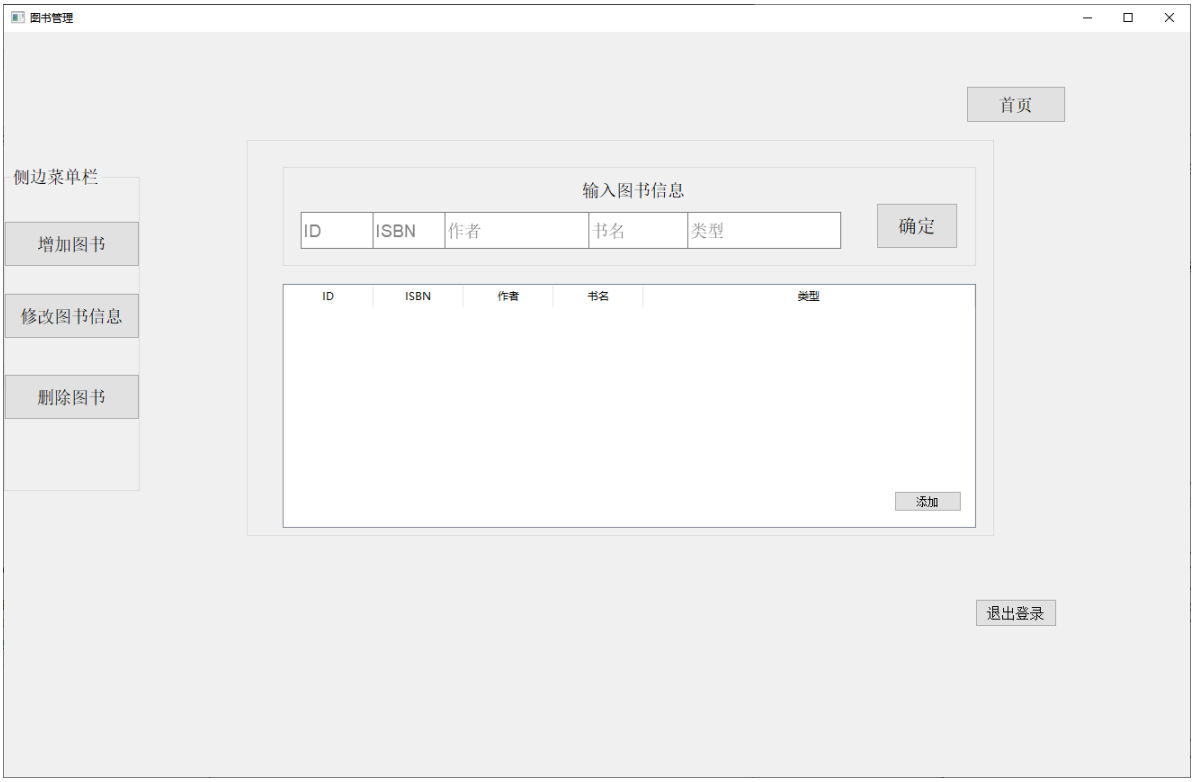

Dialog
登录
账号
密码
管理员身份
确定
退出

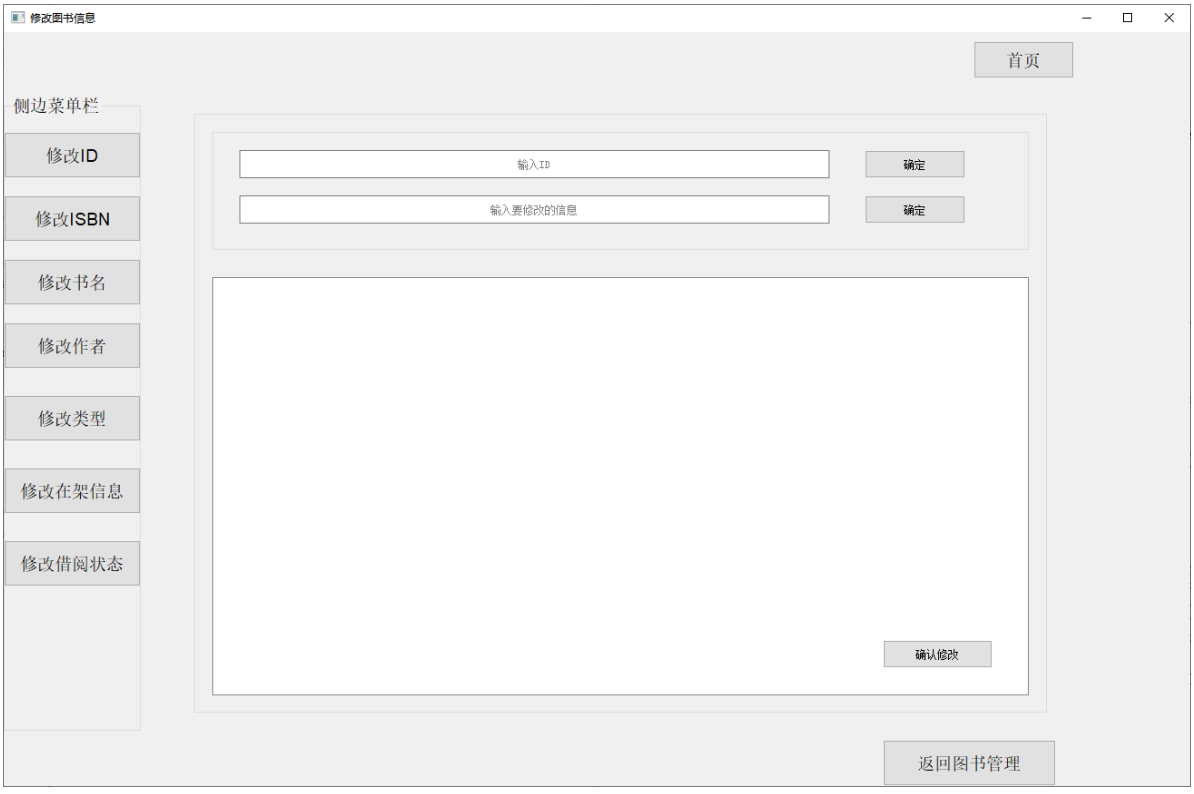

首页
用户管理
图书管理
退出登录

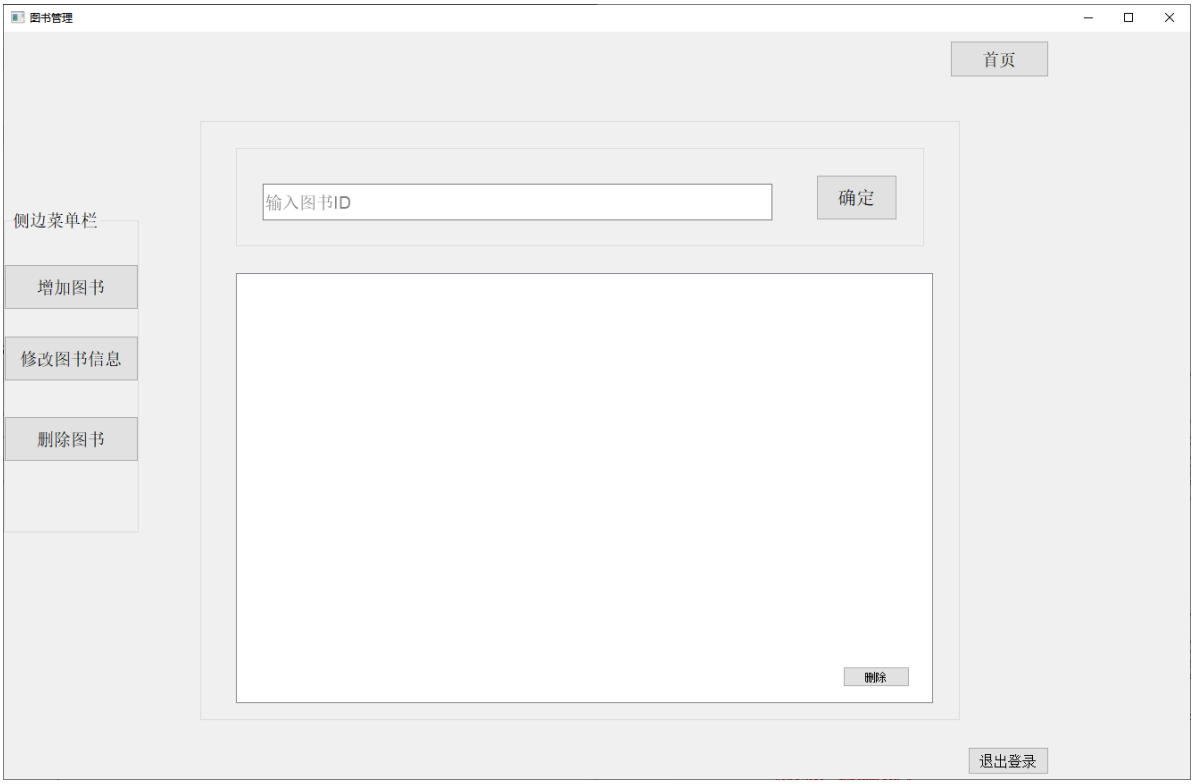

用户管理
首页
侧边菜单栏
增加用户
删除用户
修改用户
输入用户信息
ID
姓名
学校
专业
邮箱
确定
ID
姓名
学校
专业
邮箱
添加
退出登录


修改用户信息
首页
侧边菜单栏
修改ID
修改姓名
修改学校
修改专业
修改邮箱
输入ID
确定
输入要修改的信息
确定
确认修改
返回用户管理


用户管理
首页
侧边菜单栏
增加用户
删除用户
修改用户
输入学生ID
确定
删除
退出登录


图书管理
首页
侧边菜单栏
增加图书
修改图书信息
删除图书
输入图书信息
ID
ISBN
作者
书名
类型
确定
ID
ISBN
作者
书名
类型
添加
退出登录


修改图书信息
首页
侧边菜单栏
修改ID
修改ISBN
修改书名
修改作者
修改类型
修改在架信息
修改借阅状态
输入ID
确定
输入要修改的信息
确定
确认修改
返回图书管理


图书管理
首页
输入图书ID
确定
侧边菜单栏
增加图书
修改图书信息
删除图书
删除
退出登录


Form
首页
查找借阅图书
已借图书
归还图书
退出登录


MainWindow
查找借阅图书
首页
请先选择查找方式，在下方输入ISBN或ID或书名
确定
侧边菜单栏
ID查找
ISBN查找
书名查找
分类浏览
借书卡
ID
书名
ISBN
在架信息
借阅
退出登录


Form
归还图书
查找借阅图书
请在下面输入图书ID
确定
ID
书名
ISBN
作者
归还
退出登录

(2). 代码展示(Qt)

由于代码是从底层、逻辑层适配Qt而得，其实现原理不变而形式发生变化，因此只展示部分代码

```
#include "mainpage.h"
#include "ui_mainpage.h"
mainpage::mainpage(QWidget* parent) :
    QMainWindow(parent),
    ui(new Ui::mainpage)
{
    ui->setupUi(this);
}

mainpage::~mainpage()
{
    delete ui;
}
void mainpage::on_usermanagement_clicked()
{
    this->close();
    usermanager* um = new usermanager();
    um->show();
}
void mainpage::on_bookmanagement_clicked()
{
    this->close();
    bookmanager* bm = new bookmanager();
    bm->show();
}

void mainpage::on_exitlogin_clicked()
{
    this->close();
}
```