

湖南大学实验报告

课程名称	程序设计实验				
项目名称	LibrarySystem_HNU_InformationSecurity2024_2401				
项目代码量	1404 行				
完成时间	2025.1.6				
组长姓名	谢静甜	学号	202408060121	贡献比例	60%
组员姓名	李怡萱	学号	202408060123	贡献比例	40%

仓库链接: [Moyuin-aka/LibrarySystem_HNU_InformationSecurity2024_2401:group member:谢静甜, 李怡萱 \(github.com\)](#)

一、实验目的

掌握面向对象的管理信息系统开发，体验类的封装与继承

二、主要工作

该图书管理系统具有注册和登录账户（管理员——管理员由编码者注册，确
保安全性/读者）的功能。

登录成功后，普通用户将拥有“显示所有图书”“搜索图书”“借书”“还
书”的功能。考虑到有些用户不知道哪本书好看，我们增设了“今日推荐”功能，
读者便可以根据书本的受欢迎程度来借阅书本了。在“搜索图书”板块，我们也
下足了功夫，考虑到书名太长，记忆不清的问题，我们将搜索设置成了“模糊搜
索”，也就是，假设书名有“C++ Basics”“Advanced C++”，此时你只需输入
“C”，甚至是“c”，都可以显示出所有带有“c”的图书。同时，我们也提供
了按照作者或者 ISBN 码来搜索图书的途径。在“还书”功能中，考虑到需输入
所借书籍的完整书名，很是繁琐，我们增添了“一键还书”功能，提高还书效率。

这些是普通用户拥有的基本功能，而管理员在拥有以上功能的基础上，还增
添了管理图书和用户的功能，另外，管理员没有借书权限。管理图书的部分，设
有增、删、改图书信息的功能。用户管理的部分，在“查看所有用户”部分，管
理员只能看到用户名及其身份（管理员/读者），而不能查看用户所设置的密码，

确保了用户的信息安全。考虑到用户可能会忘记密码，这时可以请管理员帮忙重置密码或者删除原账号，进行重新注册。

除此之外，我们设置了“借阅排行榜”，并根据书本借阅情况，给出“十佳读者排行榜”“图书借阅排行榜”，鼓励大家读书并提供推荐书单。

三、实验收获

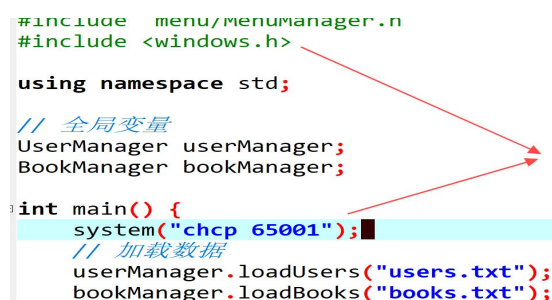
在编写该项目时，我们也遇到了困难。首先是，程序内外部分别出现了中文乱码问题。内部乱码原因是 UTF-8 编码的字符与 Dev5.11 出现中文字符 GB2312 不兼容的问题，我们使用的是 Notepad++，统一编码为 UTF-8 形式，解决了程序内部中文乱码问题，并保证了代码的兼容性，这对我们互相传递代码起到了重要的作用。外部乱码原因是在 Windows 的命令提示符（cmd）中，默认情况下是使用代码页 437（适用于美国英语），而在中文版 Windows 上，通常会使用代码页 936（GBK 编码）。

```
#include menu/MenuManager.h
#include <windows.h>

using namespace std;

// 全局变量
userManager userManager;
bookManager bookManager;

int main() {
    system("chcp 65001");
    // 加载数据
    userManager.loadUsers("users.txt");
    bookManager.loadBooks("books.txt");
}
```



我们通过添加头文件<windows.h>，并在主函数中添加 system(“chcp 65001”), 成功解决程序外部乱码问题。

后来运行时，我们发现注册账号时，账户名字中不能包含空格，若有空格，则只会读取并存储空格前内容，这是因为程序默认以空格作为输入结束标志。我们选择设置‘|’为结束标志，成功解决了该问题。

```

9      - ifstream file(filename);
10     - if (!file.is_open()) return;
11     -
12     - string title, isbn, author, publisher;
13     - double price;
14     - while (file >> title >> isbn >> author >> publisher >> price) {
15     -     books.emplace_back(title, isbn, author, publisher, price);
16     - }
17     - file.close();

10 + ifstream file(filename);
11 + if (!file.is_open()) {
12 +     cout << "无法打开文件: " << filename << endl;
13 +     return;
14 + }
15 +
16 + string line;
17 + while (getline(file, line)) {
18 +     string title, isbn, author, publisher;
19 +     double price;
20 +
21 +     // 使用字符串流解析每一行
22 +     istringstream stream(line);
23 +
24 +     getline(stream, title, '|'); // 用 "|" 作为分隔符读取图书名称
25 +     getline(stream, isbn, '|'); // 读取 ISBN
26 +     getline(stream, author, '|'); // 读取作者
27 +     getline(stream, publisher, '|'); // 读取出版社
28 +     stream >> price; // 读取价格
29 +
30 +     books.emplace_back(title, isbn, author, publisher, price);
31 + }
32 + file.close();

```

另外,我们也遇到了类中私有成员的访问问题。`private` 中的成员具有私密性,其他类无法访问,这时我们选择添加 `vector<Book>& getBooks();` 来提供访问接口,在需要用到 `books` 的时候,可以调用函数来访问。既实现了访问,又没有设置成 `public`,仍保留了数据的封装性和安全性。

```

class BookManager {
private:
    vector<Book> books;

public:
    void loadBooks(const string &filename); // 显示图书信息
    void saveBooks(const string &filename); // 保存图书信息
    void addBook(const Book &book); // 添加图书
    void deleteBook(const string &title); // 删除图书
    void updateBook(const string &title); // 修改图书
    void interactiveAddBook(); // 管理员用户交互式添加图书

    void searchByTitle(const string &title); // 按题名搜索
    void searchByAuthor(const string &author); // 按作者搜索
    void searchByIsbn(const string &isbn); // 按图书ISBN搜索
    void displayAllBooks() const; // 显示所有图书

    void borrowBook(const string &title, const string &username); // 借书
    void returnBook(const string &title, const string &username); // 还书
    void returnAllBooks(const string &username); // 一键归还用户的所有借阅图书

    vector<Book>& getBooks(); // 返回 books 的引用
    void displayBooks() const;
};

```

```

main.cpp ^ | User.h ^ | User.cpp ^ | BookManager.cpp ^ | Book.h ^ | UserManager.cpp ^ | Book.cpp
1  #ifndef MENUMANAGER_H
2  #define MENUMANAGER_H
3
4  #include "../account/AccountManager.h"
5  #include "../book/BookManager.h"
6  #include "../charts/Charts.h"
7  #include "AdminMenu.h"
8  #include "ReaderMenu.h"
9
10 class MenuManager {
11 private:
12     AccountManager &accountManager;
13     BookManager &bookManager;
14
15 public:
16     MenuManager(AccountManager &am, BookManager &bm); // 构造函数
17     void mainMenu(); // 主菜单
18 };
19
20 #endif
21
22

```

其实，在很多时候，添加功能很令人头疼，因为极易在添加代码后使得本来能够正常运行的代码出现 bug。例如，我们在添加排行榜功能时，更新 BookManager 时，需用到 UserManager。这时我们往往记得添加头文件。可是却忘了调用有接口的相关函数，导致运行出错。

```

codes/book/BookManager.h
... @@ -1,6 +1,7 @@
1 1  #ifndef BOOKMANAGER_H
2 2  #define BOOKMANAGER_H
3 3
4 4  + #include "../user/UserManager.h"
4 5  #include "Book.h"
5 6  #include <vector>
6 7  #include <string>
@@ -10,9 +11,11 @@ using namespace std;
10 11
11 12  class BookManager {
12 13  private:
14 14  +     UserManager &userManager; // 引用 UserManager
13 15     vector<Book> books;
14 16
15 17  public:
18 18  +     BookManager(UserManager &um); // 构造函数
16 19     void loadBooks(const string &filename); // 显示图书信息
17 20     void saveBooks(const string &filename); // 保存图书信息
18 21     void addBook(const Book &book); // 添加图书

```

谢静甜心得体验：

做这个图书管理系统的过程，感觉就像是搭积木，一点一点把自己心里想要的东西拼出来，最后真的完成了，挺开心的。这是我第一次独立负责一个完整的项目，从架构设计到代码实现几乎都是我一个人完成的。虽然一开始挺忐忑的，但越做越觉得有意思。

最开始，我对面向对象其实没啥概念，写过一个小动物园管理的小程序，隐约

觉得“类和对象”好像就是“把函数分到不同的文件里”。但到了这次大项目，我学着用类去划分功能，比如账户类、图书类、用户类，还把它们按模块分到不同的文件夹里。虽然一开始写头文件引用搞得我很烦，但后来发现这真的很清晰！修改 Bug 或加新功能时，根本不用翻来翻去找代码，直接找到对应的模块就行，超级方便。

说到功能设计，我加了一个“二级密码”功能，专门用来限制管理员滥用敏感权限（比如修改密码、删数据这种操作）。虽然逻辑很简单，但这是我第一次把信息安全的理念融入到代码里，做完感觉还挺有成就感的！

除了写代码，这次项目让我接触了很多新工具和知识。比如 Git，这次是我第一次用 GitHub 管理代码，学会了怎么分支、回退，保存一些尝试失败的功能。还有 UTF-8 和 GBK 的编码问题，这个让我印象很深，别人电脑跑我的代码中文全是乱码，后来用 Notepad++ 把编码改了，完美解决问题，简直太实用了！虽然我还尝试了 CMake 和 SQLite3，但有点小失败，不过我把这些都用 Git 分支保存下来了，等以后熟练了再继续搞。

总的来说，这次实验让我成长了不少。虽然代码量也就 1400 行左右，但我每一行都亲手写的，每一个细节都经过自己的思考，这种成就是抄代码得不到的。现在回头看，觉得从一开始的生疏到现在可以流畅地搭建整个系统，这个过程真的很值得。

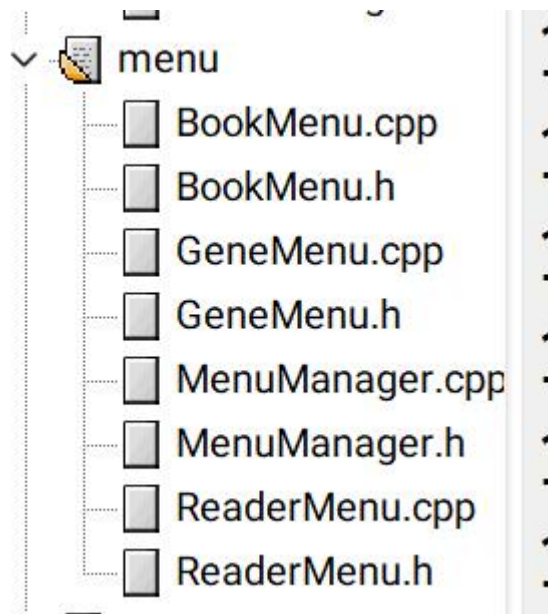
其实，编程最有意思的地方，就是你能用一行行代码把自己脑子里的想法实现出来。虽然有时候会遇到 Bug 会崩溃，但每次解决问题都能让我感受到那种“操控一切”的感觉，很爽。希望以后能做更大的项目，让代码更加出色。

李怡萱心得体验：

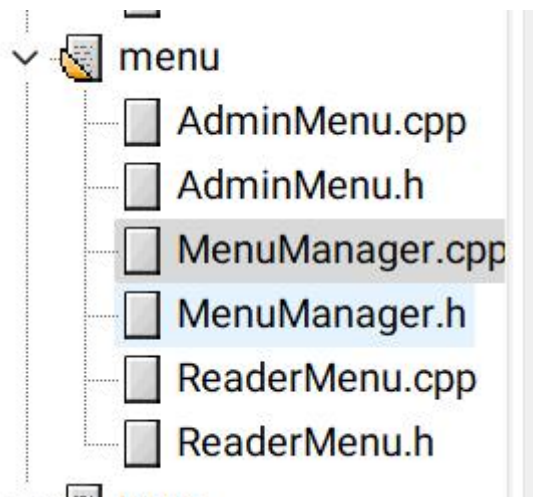
这次项目我参与程度较小。但也在和组长的交流讨论中，运行查错，功能审查，阅读代码，撰写实验报告中学到了很多知识。在接收组长写的代码时，发现在 C++ 中无法打开文件，一开始我采取了自己重新创建一个项目，复制代码的方式，最后成功运行了。后面组长使用 Notepad++ 统一了编码，使得传输编码变得简单了，此时我了解到在 C++ 打不开文件的原因是编码不同。同时自己查阅资料解决了外部乱码问题。在接收到代码后，我尝试添加功能“管理员管理用户”功能。初始想法是对 menu 栏进行调整，读取代码后，选择将 AdminMenu 改为

GeneMenu,然后增加 BookMenu。这样下来,看上去变清晰了,menu 中就含有了 MenuManager, GeneMenu, BookMenu 和 ReaderMenu。修改和添加代码后,项目不能运行了。后面,交流后发现,我 BookMenu 设想需要用到的函数在 book 栏里,我也没有理解这些折叠和代码的真正意思。后面,组长对代码进行了补充和更改。

我的思路:



组长的思路:



后面,我才逐渐理解了各个代码的功能。同时也从组长那儿学到了很多知识。比如要减少代码量,可以通过创建函数的方式,例如,管理员和读者有许多重复功能,这时采用调用函数的方式则会大大减少代码量。在交流讨论的过程中,组长向我介绍她的思路,我也提出疑问,从而一起修复了许多 bug,从中体验到了

代码的巨大乐趣。这次美中不足的地方在，编码实践少，改错能力弱，下次将带着热爱花费更多时间进行实践操作。

四、实验拓展

我们增加了“模糊搜索”和“一键还书”和“分页显示所有图书”的功能。同时，我们增加了“二级密令”。我们运用信息安全的思想，考虑到管理员权限很大，可以更改和删除用户信息，设置了二级密令，且二级密令一经设置，不可修改（只可在源代码 `admin_password.txt` 中修改，在程序运行时无法修改），实现对管理员身份的验证，保证用户信息安全。再者，在管理员重置用户密码后，用户可以设置新密码，这样也保证了用户的信息安全。

“模糊搜索”为读者搜索图书提供了便利。

```
137 - // 按题名搜索
138 - void BookManager::searchByTitle(const string &title) {
139 -     for (const auto &book : books) {
140 -         if (book.title == title) {
141 -             book.display();
142 -             return;
143 -         }
144 -     }
145 -     cout << "未找到名为 '" << title << "' 的图书。" << endl;
146 - }
147 + // 按题名搜索,支持模糊搜索
148 + void BookManager::searchByTitle(const string &titlePart) {
149 +     vector<Book> matchedBooks;
150 +
151 +     // 转换输入关键字为小写
152 +     string lowerTitlePart = titlePart;
153 +     transform(lowerTitlePart.begin(), lowerTitlePart.end(), lowerTitlePart.begin(), ::tolower);
154 +
155 +     // 搜索匹配的图书
156 +     for (const auto &book : books) {
157 +         // 转换书名为小写
158 +         string lowerTitle = book.title;
159 +         transform(lowerTitle.begin(), lowerTitle.end(), lowerTitle.begin(), ::tolower);
160 +
161 +         // 模糊匹配
162 +         if (lowerTitle.find(lowerTitlePart) != string::npos) {
163 +             matchedBooks.push_back(book);
164 +         }
165 +     }
166 +
167 +     // 如果没有找到匹配的图书
168 +     if (matchedBooks.empty()) {
169 +         cout << "未找到包含 '" << titlePart << "' 的图书。" << endl;
170 +         return;
171 +     }
172 + }
```

```
3 // 按题名搜索,支持模糊搜索
1 void BookManager::searchByTitle(const string &titlePart) {
2     vector<Book> matchedBooks;
3
4     // 转换输入关键字为小写
5     string lowerTitlePart = titlePart;
6     transform(lowerTitlePart.begin(), lowerTitlePart.end(), lowerTitlePart.begin(), ::tolower);
7
8     // 搜索匹配的图书
9     for (const auto &book : books) {
10         // 转换书名为小写
11         string lowerTitle = book.title;
12         transform(lowerTitle.begin(), lowerTitle.end(), lowerTitle.begin(), ::tolower);
13
14         // 模糊匹配
15         if (lowerTitle.find(lowerTitlePart) != string::npos) {
16             matchedBooks.push_back(book);
17         }
18     }
19 }
```


“一键还书”为还书提供便利。

```
297
298 //一键还书
299 void BookManager::returnAllBooks(const string &username) {
300     bool hasBooksToReturn = false; // 判断用户是否有需要归还的书
301     vector<string> returnedBooks; // 存储归还的书籍记录
302
303     char confirm;
304     cout << "你确定要归还所有借阅的图书吗? (y/n): ";
305     cin >> confirm;
306     if (confirm != 'y' && confirm != 'Y') {
307         cout << "已取消一键归还操作。" << endl;
308         return;
309     }
310
311     for (auto &book : books) {
312         if (book.borrower == username) {
313             book.isBorrowed = false;
314             book.borrower = "";
315             returnedBooks.push_back(book.title); // 保存归还记录
316             hasBooksToReturn = true;
317         }
318     }
319 }
```

“分页显示图书”实现了界面的美观。

```
// 分页显示, 每页最多显示 5 本图书
const int booksPerPage = 5;
int totalPages = (matchedBooks.size() + booksPerPage - 1) / booksPerPage;

for (int page = 0; page < totalPages; ++page) {
    cout << "\n==== 第 " << page + 1 << " 页, 共 " << totalPages << " 页 =====" << endl;

    for (int i = page * booksPerPage; i < (page + 1) * booksPerPage && i < matchedBooks.size(); ++i) {
        matchedBooks[i].display(); // 调用 Book 类的 display 方法显示图书信息
    }

    // 如果不是最后一页, 提示用户是否继续
    if (page < totalPages - 1) {
        char choice;
        cout << "是否继续查看下一页? (y/n): ";
        cin >> choice;
        if (choice != 'y' && choice != 'Y') {
            break;
        }
    }
}
```


“二级密令”实现用户信息安全。

```
codes/account/AccountManager.h
@@ -9,15 +9,23 @@ class AccountManager {
9   9      private:
10  10          UserManager &userManager; // 引用 UserManager
11  11          string currentUser;      // 存储当前登录用户的用户名
12  12      +      string adminSecondaryPassword; // 存储管理员二级密码
13  13
14  14      public:
15  15          AccountManager(UserManager &um); // 构造函数, 传入 UserManager
16  16          int login();                  // 登录功能
17  17          void registerAccount();      // 注册功能
18  18      +
19  19      +      void setSecondaryPassword(const std::string &password); // 设置管理员二级密码, 仅能设置一次
20  20      +      bool verifySecondaryPassword(const std::string &password) const; // 验证二级密码
21  21      +      void loadSecondaryPassword(const std::string &filename); // 保存二级密码
22  22      +      void saveSecondaryPassword(const std::string &filename) const; // 加载二级密码
23  23      +
17  24          void displayAllUsers();      // 查看所有用户
18  25          void resetPassword(const string &username); // 重置用户密码
19  26          void deleteUser(const string &username); // 删除普通用户
20  27          string getCurrentUser() const; // 获取当前登录用户
28  28      +
21  29          UserManager& getUserManager(); // 返回 UserManager 的引用
22  30      };
23  31
73  94
74  95      // 删除普通用户
75  96      void AccountManager::deleteUser(const string &username) {
76  96      -      userManager.deleteUser(username); // 调用 UserManager 方法
97  97      +      string password;
98  98      +      cout << "请输入二级密码以继续操作: ";
99  99      +      cin >> password;
100 100      +
101 101      +      if (!verifySecondaryPassword(password)) {
102 102      +          cout << "二级密码验证失败, 操作已取消!" << endl;
103 103      +          return;
104 104      +      }
105 105      +      userManager.deleteUser(username); // 调用 UserManager 方法
77 106      }
78 107
```

附言：代码量展示

