

## OOPDS HW2

學號： 113550021 姓名： 陳孟楷

### Overview

- REMINDs
- Data Structure
- Operations
- Main Functions
- Optimal Enhancement

### REMINDs

1. **加粗藍字**是通常代表的是程式碼
2. 為了編排流暢，內文中也有 optimal enhancement 的部份，**(OE)** 指的是用於 optimal enhancement 的部分

### Data Structure

- **Book**: store the information of a book, including the **title**, the published **year**, the **author** and the **number** of copies

```
class Book
{
public:
    string title;
    int year;
    string author;
    int number;

    Book(string t, int y, string a, int n)
    : title(t), year(y), author(a), number(n){}
};
```

- **Record (OE):** store the records a user or an admin has done, including the **operation**, the **title**, the **year**, and the **author** (operation only contains “**add**” a book,” **check out**” a book and “**return**” a book)

```
class Record
{
public:
    string operation, title, author;
    int year;
    Record(string o, string t, int y, string a): operation(o), title(t), year(y), author(a) {}
};
```

- **User (OE):** store the information of a user, including the **username**, the **password**, and the **record**

```
class User
{
public:
    string username, password;
    vector<Record> record;
    User(string u, string p): username(u), password(p){}
};
```

- **Admin (OE):** store the information of an admin, including the **adminname**, the **password**, and the **record**

```
class Admin
{
public:
    string adminname, password;
    vector<Record> record;
    Admin(string a, string p): adminname(a), password(p){}
```

```
};
```

- **Library**: used to control the whole library management system

```
class Library {...}; // 以下省略
```

- Inside **Library – private**, you would see the data structures below:
  - **books**: store all the books
  - **users (OE)**: store all the users
  - **admins (OE)**: store all the admins
  - **current\_user (OE)**: indicating the current user. If none, it would be **nullptr**
  - **current\_admin (OE)**: indicating the current admin. If none, it would be **nullptr**

```
vector<Book> books;  
vector<User> users;  
vector<Admin> admins;  
User *current_user = nullptr;  
Admin *current_admin = nullptr;
```

## Operations

- Add a book: input the information of a book, push into the **books** and record this operation to this admin.

```
1  
Enter Book Name: aaa  
Enter Published Year: 2025  
Enter Book Author: AAA  
Enter the Number of Available Copies: 1  
The book has been added to the system.
```

Figure 1. Screenshot of adding a book

```
void adding() // 加一本書  
{  
    string title, author;  
    int year, number;
```

```

cout << "Enter Book Name: ";
cin >> title;

cout << "Enter Published Year: ";
cin >> year;
cin.ignore();

cout << "Enter Book Author: ";
cin >> author;

cout << "Enter the Number of Available Copies: ";
cin >> number;
cin.ignore();

books.push_back(Book(title, year, author, number));
current_admin->record.push_back(Record("Add ",title,year,author));
cout << "The book has been added to the system.\n";
}

```

- Search by the year: search the book by the year, push all the books published in the specific year by **searching\_year** into **found** and print all the books.

```

2
Enter Year: 2025
Name      Author   Year      Number of Available Copies
aaa       AAA       2025      1

```

Figure 2. Screenshot of searching books by year

```

void searching() // 用年份找書
{
    cout << "Enter Year: ";
    int year;

```

```

cin >> year;
cin.ignore();
vector<Book> found = searching_year(year);
if(found.empty()) cout << "There is no book published in " << year << ".\n";
else
{
    cout << "Name\tAuthor\tYear\tNumber of Available Copies\n";
    for(auto &i : found)
        cout << i.title << "\t" << i.author << "\t" << i.year << "\t" << i.number << "\n";
}
}

```

- **searching\_year**: find the books in the specific year and return them

```

vector<Book> searching_year(int year) // 用年份去找要的书
{
    vector<Book> result;
    for(int i = 0; i < books.size(); ++i)
        if(books[i].year == year)
            result.push_back(books[i]);

    sorting_title(result); // return 之前先 sort 一下
    return result;
}

```

- **sorting\_title**: apply **insertion sort** on books to satisfy lexicographical order. It works by iterating from the second element (index 1) to the end, taking each element as a “key,” and inserting it into its correct position within the already-sorted prefix by shifting larger elements one position to the right.

```

void sorting_title(vector<Book> &result) // 用 insertion sort 排字典序

```

```

{
    for (int i = 1; i < result.size(); ++i)
    {
        Book key = result[i];
        int j = i - 1;
        while (j >= 0 && result[j].title > key.title)
        {
            result[j + 1] = result[j];
            j--;
        }
        result[j + 1] = key;
    }
}

```

- Check out a book: enter the title, decrease the number of copies if it is greater than 0, and record this operation to this user/admin.

```

3
Book Name: aaa
Borrow Successfully!

```

Figure 3. Screenshot of checking out a book

```

void checking() // 借書
{
    string title;
    cout << "Book Name: ";
    cin >> title;
    int idx = finding_book(title);
    if(idx == -1)
    {
        cout << "The Book can't be found.\n";
    }
}

```

```

        return;
    }
    else if(books[idx].number <= 0)
    {
        cout << "All books have been lent out\n";
        return;
    }
    else
    {
        books[idx].number--;
        cout << "Borrow Successfully!\n";
        if(current_user) current_user->record.push_back(Record("Check out",
books[idx].title, books[idx].year, books[idx].author));
        if(current_admin) current_admin->record.push_back(Record("Check out",
books[idx].title, books[idx].year, books[idx].author));
    }
}

```

- Return a book: enter the title, increase number of copies if the book exists, and record this operation to this admin/user.

```

4
Book Name: aaa
Return Successfully!

```

Figure 4. Screenshot of returning a book

```

void returning() // 還書
{
    string title;
    cout << "Book Name: ";
    cin >> title;
}

```

```

int idx = finding_book(title);
if(idx == -1)
{
    cout << "Can't find this book.\n";
    return;
}
else
{
    books[idx].number++;
    cout << "Return Successfully!\n";
    if(current_user) current_user->record.push_back(Record("Return ",
books[idx].title, books[idx].year, books[idx].author));
    if(current_admin) current_admin->record.push_back(Record("Return ",
books[idx].title, books[idx].year, books[idx].author));
}
}

```

- List all the books: use **sorting\_title** (already introduced above) to sort books and print them out if it's not empty.

5	Name	Author	Year	Number of Available Copies
aaa	AAA	2025	1	

Figure 5. Screenshot of listing all the books

```

void listing() // 列出所有書 一樣按照字典序
{
    if(books.empty()) cout << "There is no any book.\n";
    else
    {
        vector<Book> copy = books;

```



```

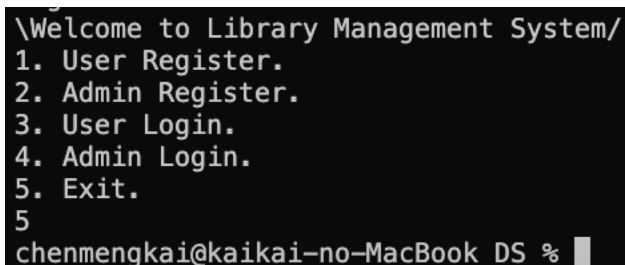
        sorting_title(copy);

        cout << "Name\tAuthor\tYear\tNumber of Available Copies\n";

        for(auto &i : copy)
            cout << i.title << "\t" << i.author << "\t" << i.year << "\t" << i.number << "\n";
    }
}

```

- Exit: if enter “5”, break the while loop and stop the system.



```

\Welcome to Library Management System/
1. User Register.
2. Admin Register.
3. User Login.
4. Admin Login.
5. Exit.
5
chenmengkai@kaikai-no-MacBook DS %

```

Figure 6. Screenshot of exiting the system

```

cout << "5. Exit.\n";
else if(num == 5) break;

```

## Main Functions

進到系統後要先註冊帳號並登入系統，並將用戶分為 user 和 admin 兩種。 (OE)  
登入後可以進行以下功能：

1. Add: 只有 admin 有權限
2. Search: user / admin 皆有權限
3. Check out: user / admin 皆有權限
4. Return: user / admin 皆有權限
5. List: user / admin 皆有權限
6. Record (OE): user / admin 皆有權限
7. Personnel management (OE): 只有 admin 有權限
8. Log out (OE): user / admin 皆有權限

[P.S.] Optimal Enhancement 功能的介紹在最後面。

```
\Welcome to Library Management System/  
1. User Register.  
2. Admin Register.  
3. User Login.  
4. Admin Login.  
5. Exit.
```

Figure 7. Login interface

```
1. Add a book.  
2. Search a book by published year.  
3. Check out a book.  
4. Return a book.  
5. List all books.  
6. Print Records.  
7. Personnel Management.  
8. Log out.
```

Figure 8. Admin Interface

```
1. Searching a book by published year.  
2. Check out a book.  
3. Return a book.  
4. List all books.  
5. Print Records.  
6. Log out.
```

Figure 9. User Interface

## Optimal Enhancement

- **registering\_user**: register with **username** and identical **passwords** (user need to enter twice) and push it into **users**

```
void registering_user() // user 註冊  
{  
    string username, password1, password2;  
    cout << "Enter Username: ";  
    cin >> username;  
    if(finding_user(username) != -1 && finding_admin(username) != -1)  
        cout << "The name has already exist. Try another.\n";  
    else  
    {
```

```

    cout << "Enter password: ";
    cin >> password1;

    while(true)
    {
        cout << "Enter the password again: ";
        cin >> password2; // 密碼要確認一次
        if(password1 == password2)
        {
            users.push_back(User(username, password1)); // 建立新用戶
            cout << "User " << username << " has been created successfully.\n";
            return;
        }
        else
            cout << "Two passwords doesn't match, please try again.\n";
    }
}
}

```

- **registering\_admin**: register with **adminname** and identical **passwords** (user need to enter twice) and push it into **admins**

```

void registering_admin() // admin 註冊
{
    string adminname, password1, password2;
    cout << "Enter Adminname: ";
    cin >> adminname;
    if(finding_user(adminname) != -1 && finding_admin(adminname) != -1)
        cout << "The name has already exist. Try another.\n";
}

```

```

else
{
    cout << "Enter password: ";
    cin >> password1;

    while(true)
    {
        cout << "Enter the password again: ";
        cin >> password2; // 密碼要確認一次
        if(password1 == password2)
        {
            admins.push_back(Admin(adminname, password1)); // 建立新 admin
            cout << "Admin " << adminname << " has been created successfully.\n";
            return;
        }
        else
            cout << "Two passwords doesn't match, please try again.\n";
    }
}
}

```

- **login\_user:** verify the user and change **current->user** into this user

```

bool login_user() // user login
{
    string username, password;
    cout << "Enter username: ";
    cin >> username;
    cout << "Enter password: ";

```

```

cin >> password;

int idx = finding_user(username); // 找看看 user 是否存在
if(idx != -1 && users[idx].password == password)
{
    current_user = &users[idx];
    cout << "Login successfully. Welcome, " << current_user->username << "!\n";
    return true;
}
else
{
    cout << "Wrong username or password\n";
    return false;
}
}

```

- **finding\_user**: find the position of a user

```

int finding_user(const string &username) // 找 user
{
    for(int i = 0; i < users.size(); ++i)
        if(users[i].username == username)
            return i;
    return -1;
}

```

- **login\_admin**: verify the admin and change **current->admin** into this admin

```

bool login_admin() // admin login
{

```

```

string adminname, password;
cout << "Enter adminname: ";
cin >> adminname;
cout << "Enter password: ";
cin >> password;
int idx = finding_admin(adminname); // 找看看 admin 是否存在
if(idx != -1 && admins[idx].password == password)
{
    current_admin = &admins[idx];
    cout << "Login successfully. Welcome, " << current_admin->adminname <<
    "\n";
    return true;
}
else
{
    cout << "Wrong adminname or password\n";
    return false;
}
}

```

- **finding\_admin**: find the position of an admin

```

int finding_admin(const string &adminname) // 找 admin
{
    for(int i = 0; i < admins.size(); ++i)
        if(admins[i].adminname == adminname)
            return i;
    return -1;
}

```

- **recording\_user**: output the record of a user

```
void recording_user() // 記錄 user operations
{
    cout << "--- User Records for " << current_user->username << " ---\n";
    cout << "Operation" << "\t" << "Title" << "\t" << "Year" << "\t" << "Author\n";
    for(auto &i : current_user->record)
        cout << i.operation << "\t" << i.title << "\t" << i.year << "\t" << i.author << "\n";
}
```

- **recording\_admin**: output the record of an admin

```
void recording_admin() // 記錄 admin 的 operations
{
    cout << "--- Admin Records for " << current_admin->adminname << " ---\n";
    cout << "Operation" << "\t" << "Title" << "\t" << "Year" << "\t" << "Author\n";
    for(auto &i : current_admin->record)
        cout << i.operation << "\t" << i.title << "\t" << i.year << "\t" << i.author << "\n";
}
```

- personnel management – **upgrade\_users**: list all the users at first, move the user from users to admins to evaluate permissions

```
void upgrade_user()
{
    cout << "Users: ";
    for(auto &u : users)
        cout << u.username << " ";
    cout << "\n";

    cout << "Promote user: ";
```

```

string name;
cin >> name;

int i = finding_user(name);
if(i < 0)
{
    cout << "No such user.\n";
    return;
}
admins.emplace_back(users[i].username, users[i].password);
users.erase(users.begin()+i);
cout << "Promoted.\n";
}

```

- personnel management – **downgrade\_admins**: list all the admins at first, move the admin from admins to users to reduce permissions

```

void downgrade_admin()
{
    cout << "Admins: ";
    for(auto &i : admins)
        cout << i.adminname << " ";
    cout << "\n";

    cout << "Demote admin: ";
    string name;
    cin >> name;

    int i = finding_admin(name);
}

```



```

    if(i < 0)
    {
        cout << "No such admin.\n";
        return;
    }
    users.emplace_back(admins[i].adminname, admins[i].password);
    admins.erase(admins.begin()+i);
    cout << "Demoted.\n";
}

```

- personnel management – **delete\_users**: delete the user and erase it in users

```

void deleting_user()
{
    cout << "Users: ";
    for(auto &u : users)
        cout << u.username << " ";
    cout << "\n";

    cout << "Delete user: ";
    string name;
    cin >> name;

    int i = finding_user(name);
    if(i < 0)
    {
        cout << "No such user.\n";
        return;
    }
}

```

```
users.erase(users.begin() + i);  
  
cout << "Deleted.\n";  
  
}
```

- log out: change the `current_user->username` or `current_admin->adminname` into `nullptr`

```
void logout_user() // user logout  
{  
    cout << "Logout: " << current_user->username << "\n";  
    current_user = nullptr;  
}  
  
void logout_admin() // admin logout  
{  
    cout << "Logout: " << current_admin->adminname << "\n";  
    current_admin = nullptr;  
}
```