

# 程序设计文档

## 目录

一、程序简介	-----1
二、变量、基础功能函数、后端文件的说明	-----1
三、程序整体框架分析	-----4
四、模块化解构流程图及函数	-----4
五、创新功能及容错性分析	-----10

### 一、程序简介

本程序为电影信息管理系统，旨在通过渲染出的图形界面和控制台相结合的方式，服务于用户的电影信息检索、票务信息管理和管理员的电影信息管理。

### 二、变量、基础功能函数、后端文件的说明

#### 1、程序中的全局变量

```
extern wchar_t *p_user;          //记录 user 的 id

void add_history(wchar_t*pointer); //用于 history 操作的记录

extern float acc;                //标记用户是否为 VIP，是 VIP 取 acc=0.95，不是则为 1

extern wchar_t *film_name;       //电影名称的记录

extern struct film               //电影信息记录的结构体
{
    int bianhao;
    wchar_t name[50];
    float ratings;
    int if_on;
};

extern struct film *p_film;       //指向该结构体的指针
```

#### 2、程序中的基础功能函数

为了更好地应对一些程序中频繁重复出现的代码，特编写了以下 4 个基础功能函数

(1) char \*cat(wchar\_t \*file, char package[100]); //文件名的连接

由于读写的相关文件包含于文件夹中，因此使用本函数进行文件夹名和文件名的连接，其中 file 存储文件名字符串的首地址，package[100]存储文件夹名并作为最后输出的路径名；函数中同时也涉及到将 wchar\_t 型的文件名转换为 char 型的过程

```
(2) wchar_t *read_file(char *p);          //文件的读写
```

由于本程序中频繁设计读取文本内容并进行细微更改的操作，因此使用本函数实现对指定文件的内容读取并存入动态数组，其中 p 存储路径字符串的首地址，输出的是动态开辟存储空间的指针

```
(3) int choose_button(int x11,int x12,int y11,int y12,int x21,int x22,int y21,int y22);  
    //选择函数
```

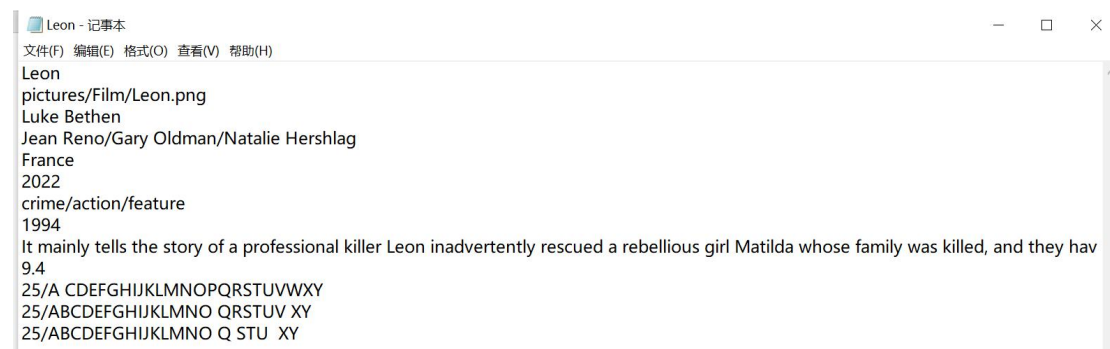
用于实现二选一判断，其中 x、y 等坐标分别标识点击范围的顶点

```
(4) int click_button(int x,int y,int x_width,int y_width);        //判断是否点击的函数
```

用于实现“CONFIRM”按键等的实现，包含 while(1)循环，只有进行点击后，程序才会继续运行，从而实现在某页面停留、检查之前操作的目的

### 3、程序中的后端文件

(1) film 文件夹：其中包含已录入的电影信息（不论上架与否），包括 Leon.txt，Inception.txt 等。其中每一个电影信息文件具有固定格式，如下图（由于电影信息曾在修改电影信息模块编程调试过程中进行随意调整，可能不完全符合实际，仅作为示例）：



各行为：电影名称

海报图片存储路径

导演

主演（使用</>作为分隔符）

拍摄国家

拍摄年份

电影类型（使用</>作为分隔符）

上映时间

简介

评分

电影院的票量设置（数字代表总票量，后加</>，之后的字母代表目前剩余的座位）

从而便于用户查看电影详细信息与购退票时进行相关信息的读取

(2) pictures 文件夹：其中 Film 文件夹存储电影海报图片，其他为系统界面所需背景图片

(3) user 文件夹：其中包含已注册的用户个人信息文件，包括 ZTR.txt 等。其中每一个用户信息文件具有固定格式，如下图：

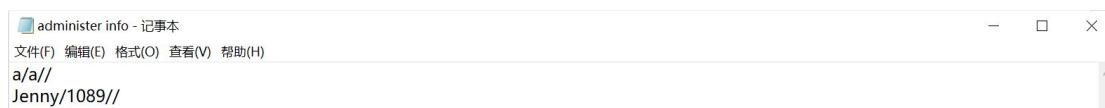


```
feature/horror/musical/feature/romance/science fiction/feature/historical/historical/historical/historical/feature/  
Leonardo DiCaprio/Jenny/ztr/  
The Legend of 1900/Leon/Schindler's List/  
[1/Leon MIDNIGHT CINEMA W]  
[1/The Legend of 1900 MIDNIGHT CINEMA E]  
[0/Leon MIDNIGHT CINEMA W]  
[1/Leon MIDNIGHT CINEMA Q]  
[1/Leon MIDNIGHT CINEMA W]  
[0/Leon MIDNIGHT CINEMA Q]
```

各行为：用户曾检索过的电影类型关键词（使用</>作为分隔符）  
用户曾检索过的电影主演关键词（使用</>作为分隔符）  
用户收藏夹（使用</>作为分隔符）  
用户购票信息（1 代表购票，0 代表退票）

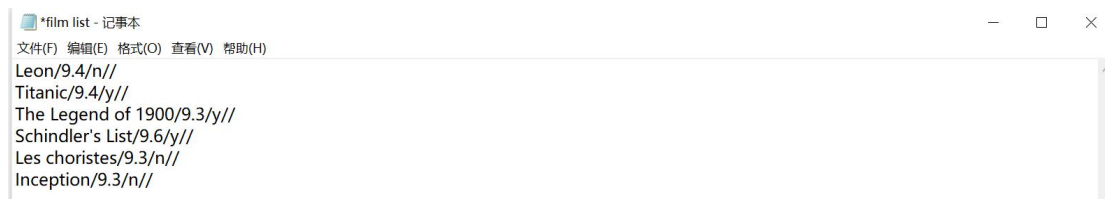
从而便于电影推荐功能以及购退票模块的编写

（4）administer info.txt：存储管理员的 id 和密码，用于管理员的登录和登出，如下图：



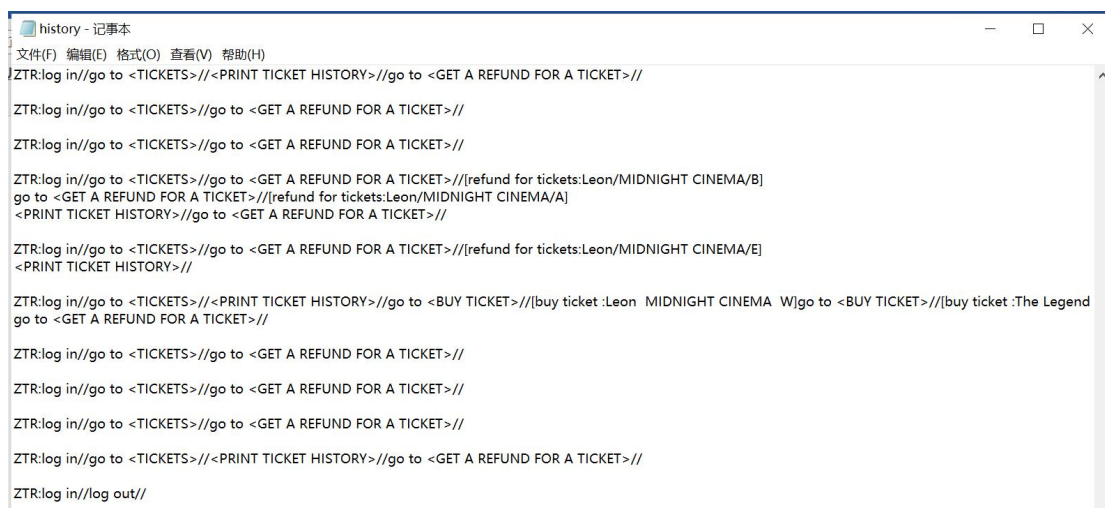
```
a/a//  
Jenny/1089//
```

（5）film list.txt：电影信息列表，格式化存储电影的“名称/评分/是否上架/”这三条最基本的信息，如下图所示，从而便于电影基本信息的读取、排序和关键词检索等功能



```
Leon/9.4/n//  
Titanic/9.4/y//  
The Legend of 1900/9.3/y//  
Schindler's List/9.6/y//  
Les choristes/9.3/n//  
Inception/9.3/n//
```

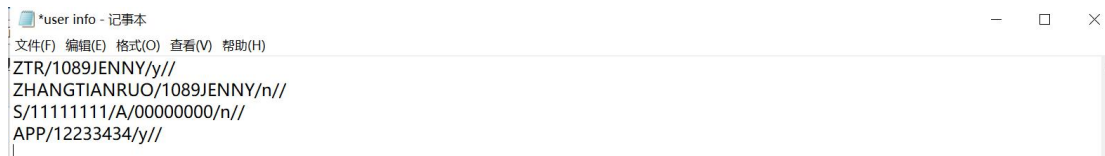
（6）history.txt：用于存储用户的各种按键操作，包括注册、登录、登出等，如下图所示：



```
ZTR:log in//go to <TICKETS>//[<PRINT TICKET HISTORY>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][go to <GET A REFUND FOR A TICKET>][refund for tickets:Leon/MIDNIGHT CINEMA/B]  
go to <GET A REFUND FOR A TICKET>][refund for tickets:Leon/MIDNIGHT CINEMA/A]  
<PRINT TICKET HISTORY>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][go to <GET A REFUND FOR A TICKET>][refund for tickets:Leon/MIDNIGHT CINEMA/E]  
<PRINT TICKET HISTORY>]  
  
ZTR:log in//go to <TICKETS>][<PRINT TICKET HISTORY>][go to <BUY TICKET>][buy ticket :Leon MIDNIGHT CINEMA W]go to <BUY TICKET>][buy ticket :The Legend  
go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//go to <TICKETS>][<PRINT TICKET HISTORY>][go to <GET A REFUND FOR A TICKET>]  
  
ZTR:log in//log out//
```

其中各信息行以登录的用户的用户名开头，后接各种标识按键或页面跳转关系的操作，当登出后另一个用户登录后，将重新开始写另一个信息行

（7）user info.txt：用于格式化存储用户的用户名、密码、是否是会员（y 代表是会员，n 代表不是会员）的基本信息，便于登录等，如下图所示：



### 三、程序整体框架分析

#### (1) 关于数据存储

没有使用现成的数据库系统，也不是通过全局数组进行存储，而是自己写了后端，通过设置了上述文本文件来模拟数据库，从而在进行相关用户、电影信息更改操作时，调用相关函数对存储该信息的文本进行读写，从而实现信息的实时更新，同时函数调用结束后，相关数据的存储空间会自行释放，从而减少程序所消耗的资源

#### (2) 关于界面的跳转

main()函数中实现点击界面中按键，跳转到其他界面完成操作再跳转回来，且程序可以始终运行的构架为：

```
int choose;
while(1)
{
    choose=page();
    int go_back=0;
    switch(choose)
    {
        case 1 : {do_action();break;}
        case 2 : {go_back=1;break;}
    }
    if(go_back==1){break;}
}
```

页面整体使用 while 循环，从而保证可以始终“回到”（其实是重新加载出）该页面，页面按键用 choose 存储，并且使用 switch 语句进行对应操作，其中 case 1 演示的是进行具体操作，case 2 演示的是如果跳出该界面，从而返回到上一个页面的方式，也即每个功能都是从初始界面开始通过线性的点击实现的

### 四、模块化解构流程图及函数

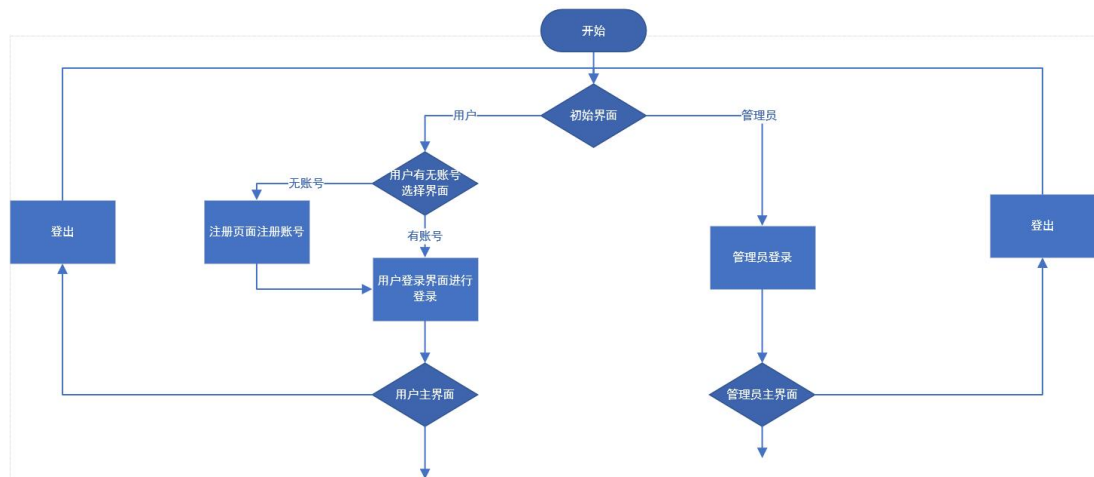
本程序主要分为用户系统和管理员系统，共同的是登录模块，其中用户系统中按界面分主要包含电影信息界面和个人信息界面，按功能分主要包含个人信息修改与查看模块、电影信息检索与查看模块、票务信息管理模块；管理员系统按界面和按功能分均分为电影信息编辑模块和用户信息查看模块

（详细运行截图请参见程序使用说明书，基本每一个步骤都进行了截图说明）

以下为分界面的程序设计思路——

#### (1) 登录界面

流程图：



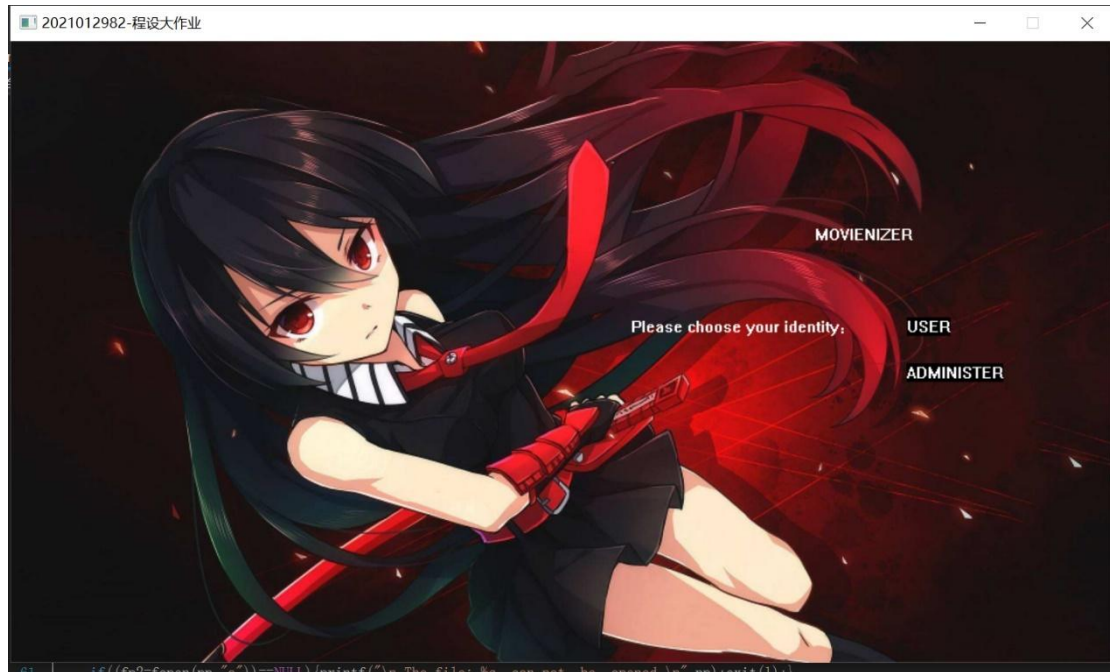
相关函数说明：

```

int login();          //初始界面中判断身份的函数
void administer_login(); //进行管理员登录的函数
int user_login();      //判断用户是否有账号的函数
void user_straightly_login(); //用户已有账号后进行用户登录的函数
wchar_t *new_user(int choose); //进行账号注册的函数
int home_page(int n);  //加载主界面的函数，其中参数 n 标记是用户/管理员身份，根据身份在界面上显示不同的按键
  
```

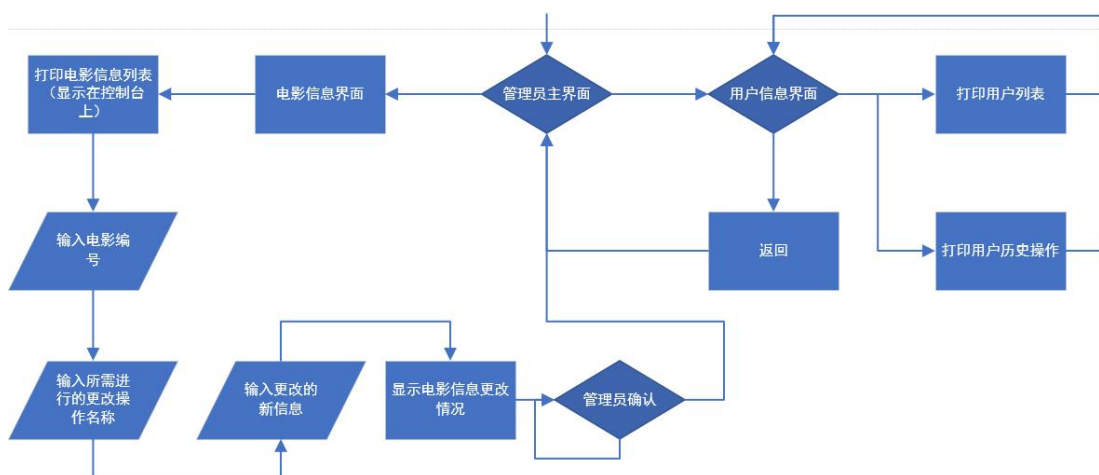
编程思路说明：

运行程序后，渲染出初始界面图片，并出现选择身份的按键，调用 login() 函数进行判断，如果是管理员，调用 administer\_login() 函数进入管理员登录界面，输入用户名、密码后调用 homepage(2) 函数进入管理员主界面；如果是用户，出现是否有账号的选择按键，调用 user\_login() 函数进行判断，没有账号者调用 new\_user(0) 函数进行注册获得账号，之后调用 user\_straightly\_login() 函数进入已有账号的用户登录界面，输入用户名和密码后调用 home\_page(1) 进入用户主界面



## (2) 管理员主界面

流程图:



相关函数说明:

### a. 电影信息编辑模块

`void edit_film_info(int n);` //管理员添加与更改电影信息的函数，主要是作为模块整体框架以及实现读取输入、显示图片上的文字、在控制台上打印文字的功能

`wchar_t *edit_details(wchar_t *point,int catt,wchar_t *newinfo);` //对电影相关信息进行后端更改的函数，即对文件文本内容进行更改，也包括调用部分特殊更改的函数

`wchar_t *add_new_film();` //添加电影的函数，主要是读入新增电影的各类信息以及后端文本的修改

`int delete_film(wchar_t *name);` //删除电影的函数，进行后端文本操作

`void film_if_on(wchar_t *name,wchar_t *newinfo);` //修改电影是否上架的函数，进行后端文本操作

`void set_ticket(wchar_t *point);` //设置票量的函数



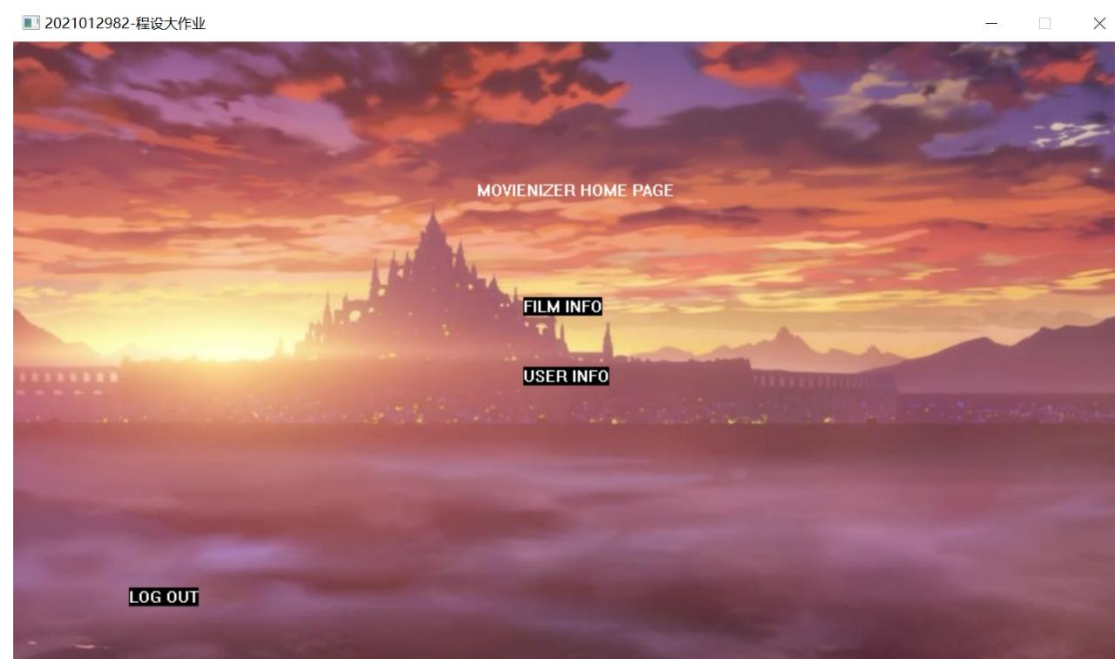
#### b. 用户信息界面

```
int user_info_page();           //用户信息界面的按键函数，即输出按了哪个按钮的标记值
void print_user_list();        //打印用户列表
void add_history(wchar_t*point); //将用户的操作全部写入 history.txt 文件中作历史记录
void print_user_history();      //打印用户的所有历史记录，包括各种按键操作
```

#### 编程思路说明：

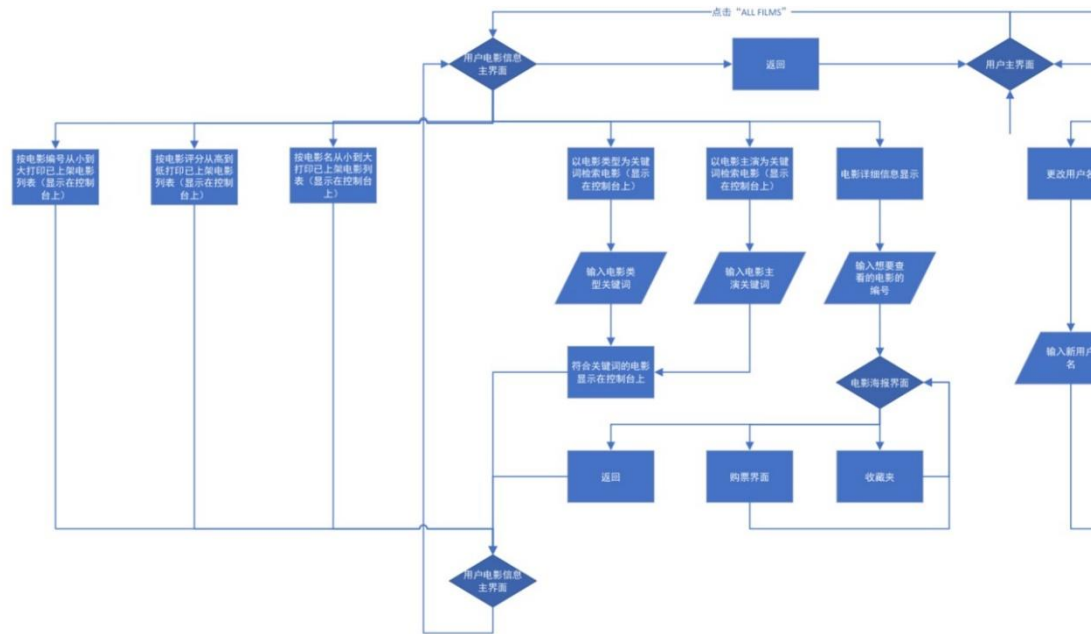
调用 homepage(2)函数进入管理员主界面后，界面上显示电影信息和用户信息两个按键，并通过 homepage(2)函数的输出值来判断点击了哪个按键，如果点击电影信息按键，则通过 edit\_film\_info()以及其函数中调用的 edit\_details()、add\_new\_film()等来实现电影信息的修改，包括电影名称、上映年份等基本信息的更改，也包括新增、删除、上下架、票量设置等，其中可供选择的操作及相关电影信息的确认会在控制台中显示；如果点击的是用户信息按键，则通过 user\_info\_page()进入用户信息界面，并通过该函数输出值判断是打印用户列表还是打印用户历史记录，这两个操作分别通过调用 print\_user\_list()和 print\_user\_history()两个函数来实现

#### 程序截图：



#### (3) 用户主界面 · 电影信息界面

#### 流程图：



(其中画了两个“用户电影信息主界面”仅仅是为了能够清楚表示在进行上述操作后将跳转回到电影信息主界面)

相关函数说明:

```

int user_film_page()    //用户的电影信息界面
int read_film_list(int judge);    //读取 film_list.txt 的内容并存入 film 结构体,并且根据函数的参数 n 分别进行按电影编号打印、按评分从高到低打印和按电影名称从小到大打印
int read_film_detailed_info(wchar_t *pointer);    //读取对应电影的 txt 文件中的详细信息
int *search_with_cat(int wherecome, wchar_t *name, int not_print);    //使用电影类型关键词进行检索, 在该部分 wherecome 取 0, name 为类型关键词, not_print 取 1
int *search_with_roles(int wherecome, wchar_t *name, int not_print);    //使用主演关键词进行检索
void add_to_favorites(wchar_t *name);    //添加到收藏夹
void recommend();    //推荐用户可能感兴趣的电影的函数
  
```

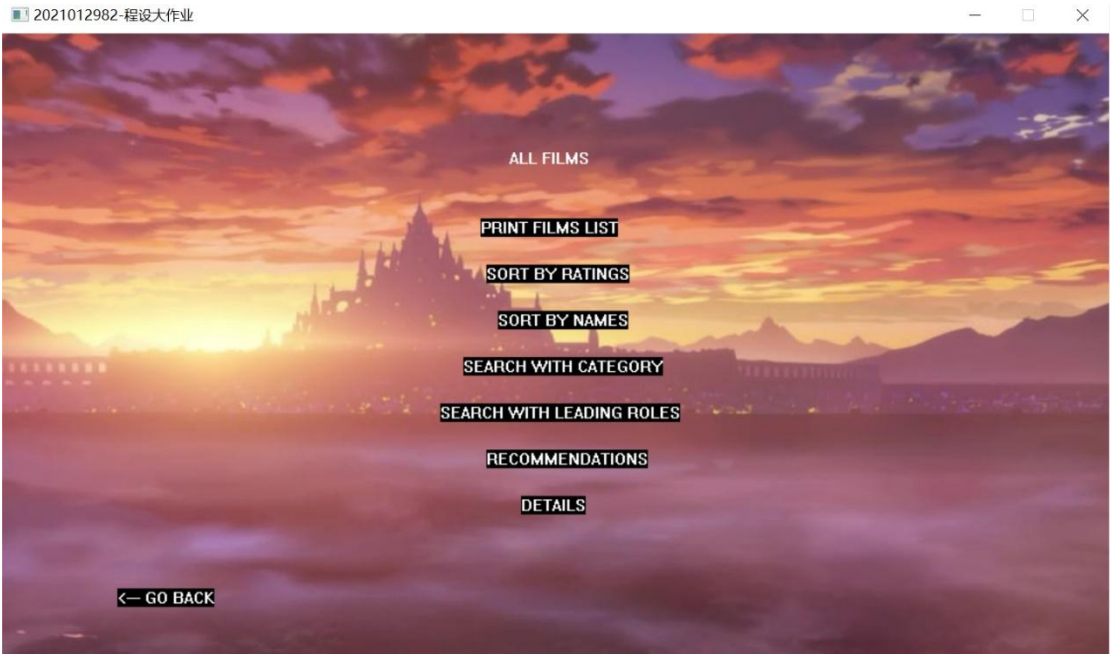
编程思路说明:

电影信息主界面上共有 8 个按键, 通过调用 user\_film\_page()函数判断按键情况, 如果点击了打印电影列表, 则调用 read\_film\_list(0)对已上架的电影列表按编号从小到大进行打印; 如果点击了按评分排序, 则调用 read\_film\_list(1)对已上架的电影的评分进行希尔排序, 按评分从高到低进行打印; 如果点击了按电影名称排序, 则调用 read\_film\_list(2)对已上架的电影的名称进行类似插入排序, 按电影名称从小到大进行打印; 如果点击了用电影类型作为关键词检索, 则调用 search\_with\_cat()函数对该关键词进行检索, 打印出所有电影类型中含该关键词的电影; 如果点击了用电影主演作为关键词检索, 则调用 search\_with\_roles()函数对该关键词进行检索, 打印出所有电影主演中含该关键词的电影; 如果点击了电影推荐, 则将调用 recommend()函数, 根据用户之前关键词检索的历史, 对这些关键词依次进行重新检索 (其中 search\_with\_cat()等函数的参数就是为了适应该过程中的一些变化而设定), 并且在检索过程中对电影被检索到的次数进行累计, 最后按照电影被检索到的次数从高到低输出推荐电影列表; 如果点击了详细信息, 则调用 read\_film\_detailed\_info()函数, 通过阅读该电影的文本

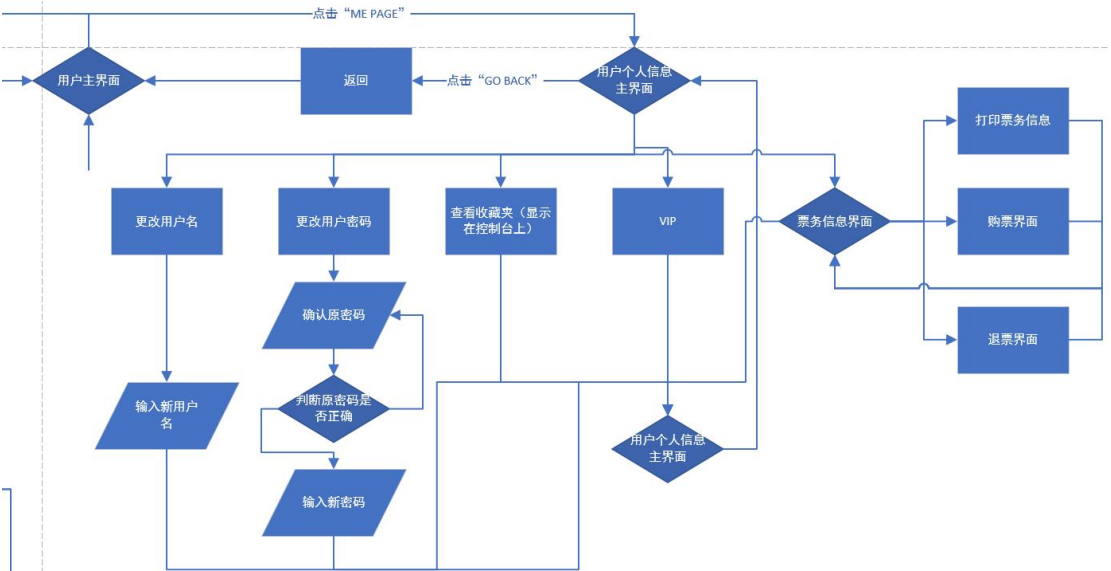


文件，打印出该电影的详细信息并展现电影海报，其中海报上有添加到收藏夹、购票的功能按键，其中购票和后文中的票务信息模块基本一致，添加到收藏夹则是通过调用 `add_to_favorites()` 函数来实现后端文件的一些操作，也即将该电影名称写入该用户的文本文件中的收藏夹对应行，便于查找和打印；点击返回按键则回到用户主界面

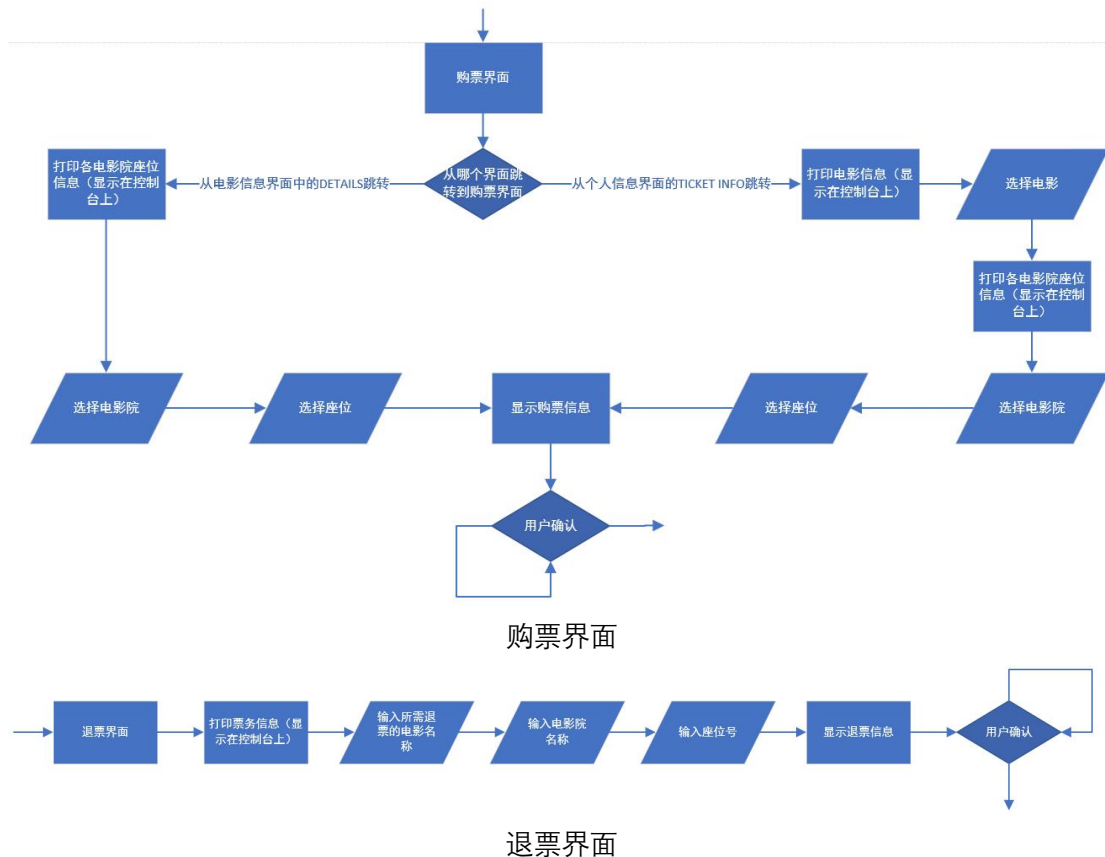
程序截图：



(4) 用户主界面 · 个人信息界面



(其中画了两个“用户个人信息主界面”仅仅是为了能够清楚表示在进行上述操作后将跳转回到个人信息主界面，购票界面和退票界面单独绘制如下)



相关函数说明：

```

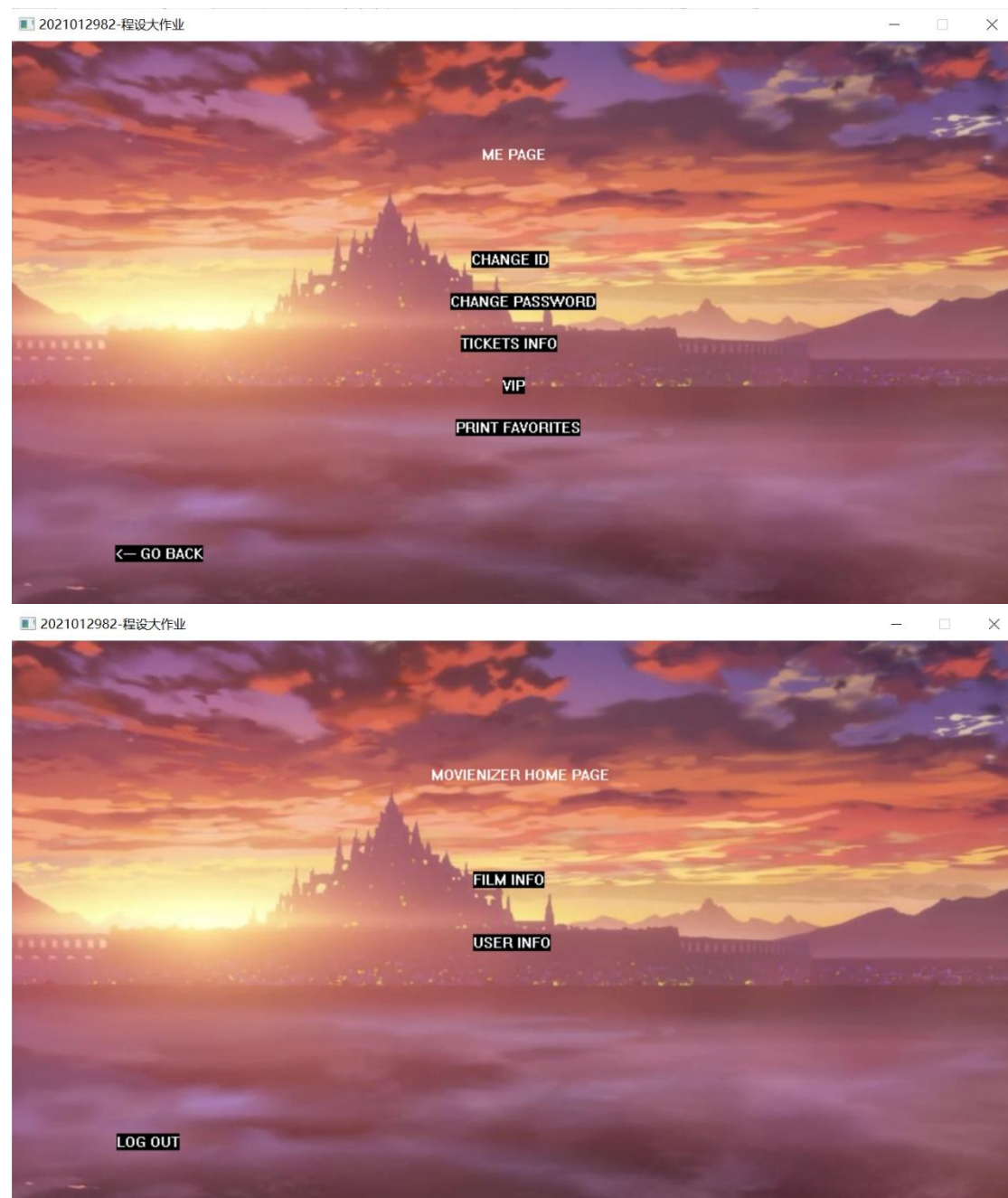
int me_page();    //user 个人页面
void user_change_id();    //用户更改用户名的函数
void user_change_pswd();  //用户更改密码的函数
void vip();        //vip 页面，可以申请成为 VIP
void print_favorites();  //查看收藏夹的函数
void ticket_page(wchar_t *pointer,int wherecome); //用于购票的函数
void print_ticket_info(int wherecome); //打印票务信息的函数，设置参数是为了与其他功能
对该函数的部分调用进行区别
int user_ticket_info_page();    //判断票务信息界面用户按键情况的函数
void ticket_refund();    //用户操作退票的函数
  
```

编程思路说明：

调用 me\_page() 函数来判断用户个人信息界面的按键情况，如果点击更改 id，则调用 user\_change\_id() 函数，将新的用户名在后端文件中进行更改，以及程序运行本身用户名标记需要更新，该函数新创建用户名的主体部分来源于对 new\_user() 函数的调用；如果点击更改密码，则调用 user\_change\_pswd() 函数，其主体部分同样来源于对 new\_user() 函数的调用；如果点击 VIP，则调用 vip() 函数进入 VIP 申请页面，已经是 VIP 的将在页面上进行语句标识，不是的可以选择在该页面上选择成为 VIP；如果点击打印收藏夹，则调用 print\_favorites() 函数在控制台上打印出用户加入收藏夹的电影名称；如果票务信息界面，将跳转进入该界面。该界面通过调用 user\_ticket\_info\_page() 函数来判断按键情况，如果点击打印票务信息，则调用 print\_ticket\_info(0) 函数可以将票务信息打印在控制台上；如果点击购票，则调用 ticket\_page() 函数实现购票，即读取用户输入的电影名、电影院名、座位，从而在后端相关

文件操作后显示购票成功界面，其上有所购票的基本信息；如果点击退票，则调用 ticket\_refund() 函数实现退票，同样是读取电影票具体信息并在后端文件上进行操作

程序截图：



## 五、创新功能及容错性分析

(1) 创新功能：使用了 EasyX 图形库，能够 a. 渲染出带有背景图片和标题的界面，并实现界面之间的合理跳转，b. 实现了鼠标消息的接收从而能够在界面上点击按键即可调用相关函数实现对应功能；开辟了收藏夹功能，并且加入收藏夹后可以在之后每一次打开对应海报界面后通过文本查找在界面上显示已加入

(2) 容错性：设置用户名时会查重，不允许出现重复用户名，且用户名长度不超过 20 个字

符；设置密码时要求密码是不少于 8 个字符的，并且要求二次输入以密码；当查找电影、购票时会对部分输入进行检查，如果电影编号存在、影院序号不存在等将报错；当编辑电影信息时会对操作名称、影院编号等进行一定检查，尽量使输入的数据是合法合理的；不一一赘述