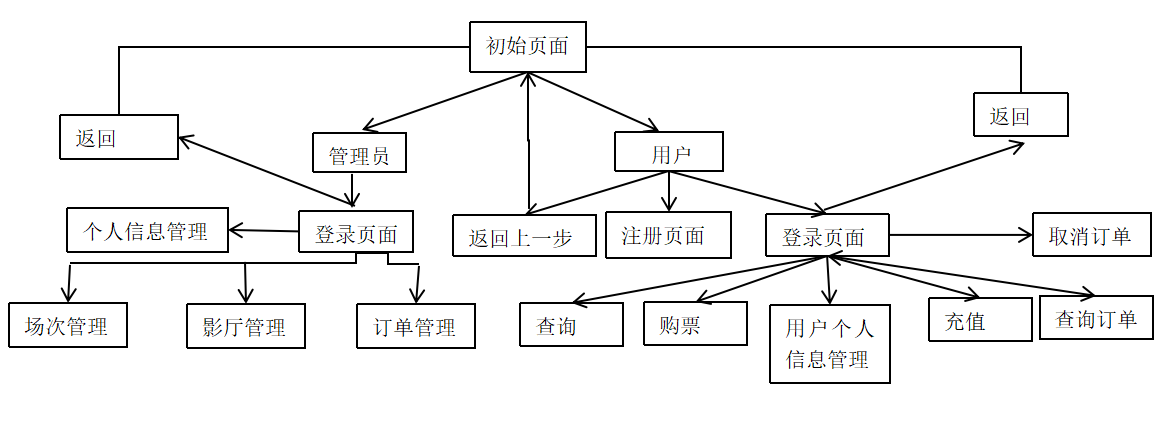
电影院销售系统设计报告

系统总体设计：

1. 系统总功能描述图：



**管理员管理**：每个管理员只能管理自己所对应的影院，能够管理的项目为订单管理，影厅管理，场次管理和个人信息管理。订单管理分为查看用户订单情况和统计用户订单情况两个小部分，查看用户订单情况显示所管理影院已售出的全部订单详细信息，统计用户订单情况分为四个小部分，按场次订单个数，场次上座率，场次总票价，影厅总票价排序。影厅管理分为影厅查询和添加影厅两个部分，影厅查询即显示所管理影院的所有影厅信息，添加影厅通过输入新的影厅的信息来完成添加工作。场次管理分为场次查询和添加场次两个部分，场次信息即为查询管理员所管理影院的所有场次的查询，添加场次即为添加新的场次。个人信息管理便是可以修改管理员的密码和邮箱，能够更加科学的使用这些信息。

**用户管理**：顾客可以选择登陆还是注册，如果选择注册，注册结束又返回登陆页面，顾客能够使用的部分有查询，购票，用户个人信息管理，充值，查询订单和取消订单，查询可分为按电影名，电影名加影院名，3D2D类型，国语英语类型来进行查询，也可通过放映开始时间，场次票价，场次余票数来查询排序结果。购票即通过各类筛选信息，通过推荐座位或自主选择座位的方式，让用户能够自由选择座位，生成订单。用户个人信息管理可以对用户的手机号，密码和邮箱进行修改。查询订单对用户所购买的订单进行查询，取消订单即可输入你要取消的订单号，然后该条订单能被取消。

1. 接口定义规范：
2. 接口定义规范：在接口部位，我们采用switch语句和多重if语句，for语句进行选择排除。在检错机制中，我们采用的是for循环和if语句进行接口。该几种接口方式保证了多种选项可行，可以通过各自的需求和选择进入不同的界面进行操作。

在进入界面时，我们将人群分为管理员和用户两类人，并分别具有不同的操作选项。管理员分为订单管理，影厅管理，场次管理和个人信息管理等；顾客分为查询，购票，用户个人信息管理，充值，查询订单和取消订单等。保证选择枝干清晰易懂，方便快捷。

1. 数据结构描述：

在数据结构部分，我们用了10个类型的数据结构，order,SES,BUY,movie,ADM,cinema,GESHU,LV,total,YT.struct orderstruct order该十个数据结构均存储在以txt形式的文件当中，可针对文件进行读取，反馈至程序当中。

当程序读文件时，将信息录入至结构，在各种选项分支中可直接利用结构的信息进行比较，筛选，条件确定和是非判断。

{

char yonghu[20];//已购票用户ID

char chang[20];//购买场次号

char zuowei[20];//座位号码

int duoshao;//购买票数

int year;//购票月

int month;//购票日

int day;//购票时

float price;//票价

char st[20];//开始时间

char et[20];//结束时间

char yingyuan[20];//影院

int numorder;//订单号

char movie[20];//电影名称

};

struct order ode[99];//订单结构

struct SES

{

char number[20];//场次号

char name[99];//电影名称

char place[100];//所属影院

int screen;//放映影厅

char start\_time[20];//电影开始时间

char end\_time[20];//电影结束时间

char totaltime[20];//电影时长

int margin;//场次余票数

float ticket;//场次票价

char kind[20];//场次放映类型

char sort[20]; //放映影厅类型

float discounts;//场次优惠活动

};

struct SES cin[99];//场次信息结构

struct BUY

{

char ID[20];//用户ID

char name[20];//姓名

char gender[20];//性别

char tele[20];//电话

char password[20];//密码

char mail[20];//邮箱

int balance;//账户余额

}buy[100];

struct BUY user[99];//用户信息结构

using namespace std;

struct movie

{

char information[12][20];

};//查询结构

struct ADM

{

char ID[10];//管理员ID

char place[20];//所属影院

char name[20];//姓名

char mail[99];//邮箱

char password[16];//密码

};

struct ADM admin[99];//管理员信息结构

struct cinema

{

int ID;//影厅编号

char ying[20];//所属影院

char seat[20];//影厅总座位

char kind[20];//影厅类型

int hang;//影厅有多少行

int lie;//影厅有多少列

};

struct cinema hall[99];//影厅结构

struct GESHU

{

char chang[20];//场次号

int num;//场次订单数

};

struct GESHU geshu[10];//场次与对应订单数

struct LV

{

char chang[20];//场次号

float rate;//场次上座率

};

struct LV lv[4];//场次与对应上座率

struct TOTAL

{

char chang[20];//场次号

float money;//场次总票价

};

struct TOTAL total[4];//场次与对应总票价

struct YT

{

int ting;//影厅号

float money;//影厅总票价

};

struct YT yt[4];//影厅与对应总票价

{

char yonghu[20];//已购票用户ID

char chang[20];//购买场次号

char zuowei[20];//座位号码

int duoshao;//购买票数

int year;//购票月

int month;//购票日

int day;//购票时

float price;//票价

char st[20];//开始时间

char et[20];//结束时间

char yingyuan[20];//影院

int numorder;//订单号

char movie[20];//电影名称

};

struct order ode[99];//订单结构

struct SES

{

char number[20];//场次号

char name[99];//电影名称

char place[100];//所属影院

int screen;//放映影厅

char start\_time[20];//电影开始时间

char end\_time[20];//电影结束时间

char totaltime[20];//电影时长

int margin;//场次余票数

float ticket;//场次票价

char kind[20];//场次放映类型

char sort[20]; //放映影厅类型

float discounts;//场次优惠活动

};

struct SES cin[99];//场次信息结构

struct BUY

{

char ID[20];//用户ID

char name[20];//姓名

char gender[20];//性别

char tele[20];//电话

char password[20];//密码

char mail[20];//邮箱

int balance;//账户余额

}buy[100];

struct BUY user[99];//用户信息结构

using namespace std;

struct movie

{

char information[12][20];

};//查询结构

struct ADM

{

char ID[10];//管理员ID

char place[20];//所属影院

char name[20];//姓名

char mail[99];//邮箱

char password[16];//密码

};

struct ADM admin[99];//管理员信息结构

struct cinema

{

int ID;//影厅编号

char ying[20];//所属影院

char seat[20];//影厅总座位

char kind[20];//影厅类型

int hang;//影厅有多少行

int lie;//影厅有多少列

};

struct cinema hall[99];//影厅结构

struct GESHU

{

char chang[20];//场次号

int num;//场次订单数

};

struct GESHU geshu[10];//场次与对应订单数

struct LV

{

char chang[20];//场次号

float rate;//场次上座率

};

struct LV lv[4];//场次与对应上座率

struct TOTAL

{

char chang[20];//场次号

float money;//场次总票价

};

struct TOTAL total[4];//场次与对应总票价

struct YT

{

int ting;//影厅号

float money;//影厅总票价

};

struct YT yt[4];//影厅与对应总票价

struct order

{

char yonghu[20];已购票用户ID

char chang[20];购买场次号

char zuowei[20];座位号码

int duoshao;购买票数

int year;购票月

int month;购票日

int day;购票时

float price;票价

char st[20];开始时间

char et[20];结束时间

char yingyuan[20];影院

int numorder;订单号

char movie[20];电影名称

};

struct order ode[99];订单结构

struct SES

{

char number[20];场次号

char name[99];电影名称

char place[100];所属影院

int screen;放映影厅

char start\_time[20];电影开始时间

char end\_time[20];电影结束时间

char totaltime[20];电影时长

int margin;场次余票数

float ticket;场次票价

char kind[20];场次放映类型

char sort[20]; 放映影厅类型

float discounts;场次优惠活动

};

struct SES cin[99];场次信息结构

struct BUY

{

char ID[20];用户ID

char name[20];姓名

char gender[20];性别

char tele[20];电话

char password[20];密码

char mail[20];邮箱

int balance;账户余额

}buy[100];

struct BUY user[99];用户信息结构

using namespace std;

struct movie

{

char information[12][20];

};查询结构

struct ADM

{

char ID[10];管理员ID

char place[20];所属影院

char name[20];姓名

char mail[99];邮箱

char password[16];密码

};

struct ADM admin[99];管理员信息结构

struct cinema

{

int ID;影厅编号

char ying[20];所属影院

char seat[20];影厅总座位

char kind[20];影厅类型

int hang;影厅有多少行

int lie;影厅有多少列

};

struct cinema hall[99]影厅结构

struct GESHU

{

char chang[20];场次号

int num/场次订单数

};

struct GESHU geshu[10];场次与对应订单数

struct LV

{

char chang[20];场次号

float rate;场次上座率

};

struct LV lv[4];场次与对应上座率

struct TOTAL

{

char chang[20];场次号

float money;场次总票价

};

struct TOTAL total[4];场次与对应总票价

struct YT

{

int ting;影厅号

float money;影厅总票价

};

struct YT yt[4];影厅与对应总票价

1. 系统开发环境：
2. C-Free是一款C/C++集成开发环境（IDE）。目前有两个版本，收费的C-Free 5.0 专业版和免费的C-Free 4.0 标准版。
3. Free中集成了C/C++代码解析器，能够实时解析代码，并且在编写的过程中给出智能的提示。C-Free提供了对目前业界主流C/C++编译器的支持，你可以在C-Free中轻松切换编译器。可定制的快捷键、外部工具以及外部帮助文档，使你在编写代码时得心应手。完善的工程/工程组管理使你能够方便的管理自己的代码。
4. txt文件：简单的文档文件类型。Windows10自带的文件格式。
5. 《标准C语言基础教程》--------------------------电子工业出版社第四版

模块详细设计：

用户：

登陆

注册

用户

密码

ID

充值

密码

邮箱

联系电话话话

性别

姓名

用户ID

**用户注册**

该模块使用函数void uregister() ，该部分运用文件userinformation.txt, 将注册好的信息写入文件中，首先用户必须按要求输入所需信息，其次对输入不符的信息，系统会检测并提示，只有当用户正确输入全部所需信息时，注册完成。

**ID**

账号为由数字和字母组合成字符串，长度为6位到10位,通过if(strlen(userID)<6||strlen(userID)>10) 语句来检查账号的长度；通过if(isalpha(userID[i]) == 0 && isdigit(userID[i]) == 0) 语句来检验既不是英文字母又不是数字的符号；当注册的帐户名称与已有的账户名称重复时，要求用户重新输入ID。

**姓名**

用户需要提供其姓名，对姓名无限制。

**性别**

用户只能按指定要求输入'man'or'woman'，如果输入有误，则要求重新输入。

**联系电话**

用户必须输入11位电话号码，对于输入不是数字、输入数字不等于11位、输入含非法字符的情况，系统会显示其输入有误，并指出其出错类型，要求用户再次输入，直到输入正确为止。

**密码**

要求用户输入2次密码，只有2次的密码完全相同，才能采用此次输入的密码。否则重新输入密码，直到2次输入相同。

**邮箱**

邮箱的格式为（\*\*\*@\*\*\*.\*\*\*），用户必须按照指定格式输入，@前后必须有内容，‘· ’前后必须要有输入内容，通过if语句来筛选判断，邮箱不符合要求，要求重新输入，直到输入正确为止。

**充值**

在注册的最后一步就是充值，系统询问用户是否充值，如果用户选择不充值，则账户余额为0，如果用户选择充值，则输入想要充值的金额，如果输入字母或者其他字符，或者输入负数，系统会自动提醒，要求其重新输入，直到输入正确为止。

在以上信息都输入完成后，系统会将用户所填信息按照声明的结构形式，写入文件中，并显示注册完成，用户科按任意键返回主菜单。

**用户登陆**

首先在enter()函数中 ,用户选择用户登录选项，要求用户输入ID与密码，验证用户输入的用户ID和密码是否正确，不正确则提示用户重新输入，共有3次输入验证机会，如果输错3次，则不能再输，返回到主页面，正确则调用userfunction()函数，进入成功登录界面。

**用户功能**

根据影片名查询

查询

根据影片名+影院名查询

用放映影厅类型3D、2D查询

用

户

功

能

用放映场次类型英语、国语查询

购票

根据放映开始时间排序所有场次

用户个人信息管理

根据场次票价排序所有场次

根据场次余票数排序所有场次

充值

查询订单

取消订单

返回

**查询**

在该界面，用户可以选择查询电影信息，包括1.根据影片名查询 2.根据影片名+影院名查询3.用放映影厅类型3D、2D查询 4.用放映场次类型英语、国语查询 5.根据放映开始时间排序所有场次 6.根据场次票价排序所有场次 7.根据场次余票数排序所有场次。

用户选择1，调用search\_fn()函数；

用户选择2，调用 selectkey()函数；

用户选择3，调用choose\_sort()函数；

用户选择4，调用choose\_kind()函数；

用户选择5，调用search\_time()函数；

用户选择6，调用search\_price()函数；

用户选择7，调用search\_margin()函数；

**search\_fn()**

提示用户输入电影名称，在for 循环中比对“场次信息.txt”中的电影名称，如果有输入的电影，则将该电影的信息：场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣，打印到屏幕上。如果输入有误，则系统提示重新输入。返回主页面请输入 1 购票请输入 2，通过if语句将输入的内容进行检查，只能时1或2。

**selectkey()**

提示用户输入电影名称和影院名，在for 循环中比对“场次信息.txt”中的电影名称和影院名称，如果有输入的电影和相对应的影院，则将该电影的信息：场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣，打印到屏幕上。如果输入有误，则系统提示重新输入。 返回主页面请输入 1 购票请输入 2，通过if语句将输入的内容进行检查，只能时1或2。

**choose\_sort()**

提示用户输入选2D还是3D电影，在for 循环中比对“场次信息.txt”中的电影类型，如果有所需的电影，则将该电影的信息：场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣，打印到屏幕上。如果输入有误，则系统提示重新输入。返回主页面请输入 1 购票请输入 2，通过if语句将输入的内容进行检查，只能时1或2。

**choose\_kind()**

提示用户输入选英语还是国语电影，在for 循环中比对“场次信息.txt”中的电影类型，如果有所需的电影，则将该电影的信息：场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣，打印到屏幕上。如果输入有误，则系统提示重新输入。返回主页面请输入 1 购票请输入 2，通过if语句将输入的内容进行检查，只能时1或2。

**search\_price()**

用户选择根据票价排序时，打开“场次信息.txt”文件，通过结构cin来找到对应的票价，然后通过冒泡排序法，两两比对，将大小换位，存入结构，然后将排好序的结构打印出来，得到所需的电影信息：场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣，并且根据票价的多少，由大到小排序。返回主页面请输入 1 购票请输入 2，通过if语句将输入的内容进行检查，只能时1或2。

**search\_margin()**

用户选择根据余票数排序所有场次时，打开“场次信息.txt”文件，通过结构cin来找到对应的余票数，然后通过冒泡排序法，两两比对，将大小换位，存入结构，然后将排好序的结构打印出来，得到所需的电影信息：场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣，并且根据余票数的多少，由大到小排序。返回主页面请输入 1 购票请输入 2，通过if语句将输入的内容进行检查，只能时1或2。

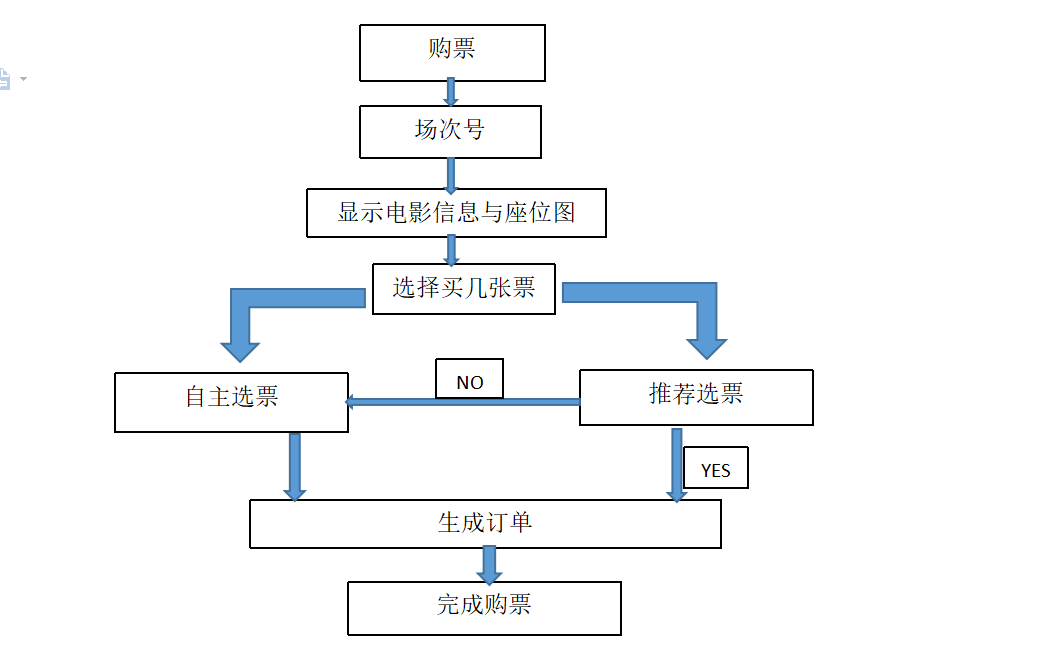
**search\_time()**

用户选择根据开始时间排序所有场次时，打开“场次信息.txt”文件，通过结构cin来找到对应的开始时间，然后通过冒泡排序法，两两比对，将大小换位，存入结构，然后将排好序的结构打印出来，得到所需的电影信息：场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣，并且根据开始时间的多少，由大到小排序。返回主页面请输入 1 购票请输入 2，通过if语句将输入的内容进行检查，只能时1或2。

查询完成后，可以选择购票进首相入goupiao()函数或者返回主页面userfunction()

购票

goupiao()



输入场次号在“场次信息.txt”中，找到相对应的电影，显示有关电影的场次号|电影名称|所属影院|放映影厅|电影开始时间|电影结束时间|电影时长|余票数|票价|影厅类型|场次类型|折扣的信息，并且显示座位图，输入所需电影票数，每个用户只能购买3张电影票，否则不能购买。

**test\_same(userID,cin[i].name)**

检测所买电影是否与订单中购买过的电影相同，如果相同则给出提示信息。让用户决定是否继续购买。

座位图采用二维数组的方式，最初都标记为0；然后读取文件的信息，将订单信息中购买本场此的座位找到，并将其标记为1，这样二维数组中，有人的标记为1，无人的标记为0，并通过printf函数，用特殊图形表示出来，更加具有直观性。

接下来用户可以选择自主选座和推荐选座。

1. 自主选票 2.推荐选票

**zizhu1(a)**

得到购买1张票，进入zizhu1函数，该函数可让用户自己输入想要的行和列，但是不能选择已经有人的座位和与有人的座位只空一个座位。这个通过比对二维数组中的0或1来实现。

**zizhu2(a)**

得到购买2张票，进入zizhu2函数，该函数可让用户自己输入想要的行和列，但是不能选择已经有人的座位和与有人的座位只空一个座位。这个通过比对二维数组中的0或1来实现。

**zizhu3(a)**

得到购买3张票，进入zizhu3函数，该函数可让用户自己输入想要的行和列，但是不能选择已经有人的座位和与有人的座位只空一个座位。这个通过比对二维数组中的0或1来实现。

**tuijian1（a）**

推荐1座位我通过计算出最佳位置的方式如：在8\*8的电影院，（i-4)的平方+（j-4）的平方最小，即为最佳座位，如果有人，则继续向外寻找，找到第二个靠近中心的座位，如果用户想要自主选票，则跳转至zizhu1(a),否则则选定推荐座位。

**tuijian2（a）**

推荐2座位我通过计算出最佳位置的方式如：在8\*8的电影院，（i-4)的平方+（j-4）的平方最小，即为最佳座位，如果有人，则继续向外寻找，找到第二个靠近中心的座位，并且比对旁边座位，若有人，则寻找下一个座位，如果用户想要自主选票，则跳转至zizhu2(a),否则则选定推荐座位。

**tuijian3（a）**

推荐3座位我通过计算出最佳位置的方式如：在8\*8的电影院，（i-4)的平方+（j-4）的平方最小，即为最佳座位，如果有人，则继续向外寻找，找到第二个靠近中心的座位，并且比对旁边2个座位，若有人，则寻找下一个座位，如果用户想要自主选票，则跳转至zizhu3(a),否则则选定推荐座位。

将用户选择座位标记为1，写入二维数组，再次展示座位图

**test\_shuliang(userID)**

这个函数通过接收用户ID,在订单信息中查找该用户完成的订单个数，如果超过5个订单，则不能购票，提示一个用户一天最多购买五个订单，您今日已达到最大额度无法继续购买，跳转至userfanction()函数。

**test\_time(char getid[20],char chang[20])**

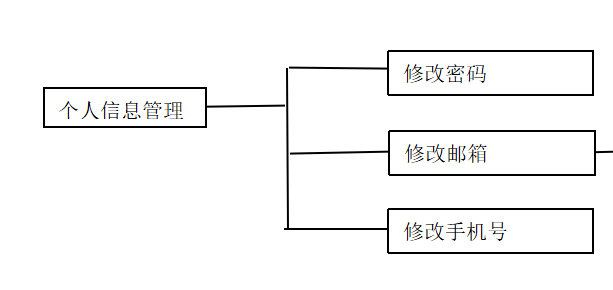
这个函数通过接收用户ID,在订单信息中查找该用户完成的订单时间，如果此次购买电影与以购买的电影时间发生冲突，则提示您购买的影片场次的时间和订单号为 这个场次时间冲突，请选择是否继续购买，购买则继续，不购买则退出。

**time\_t timer;**

获取电脑时间，生成下单时间。

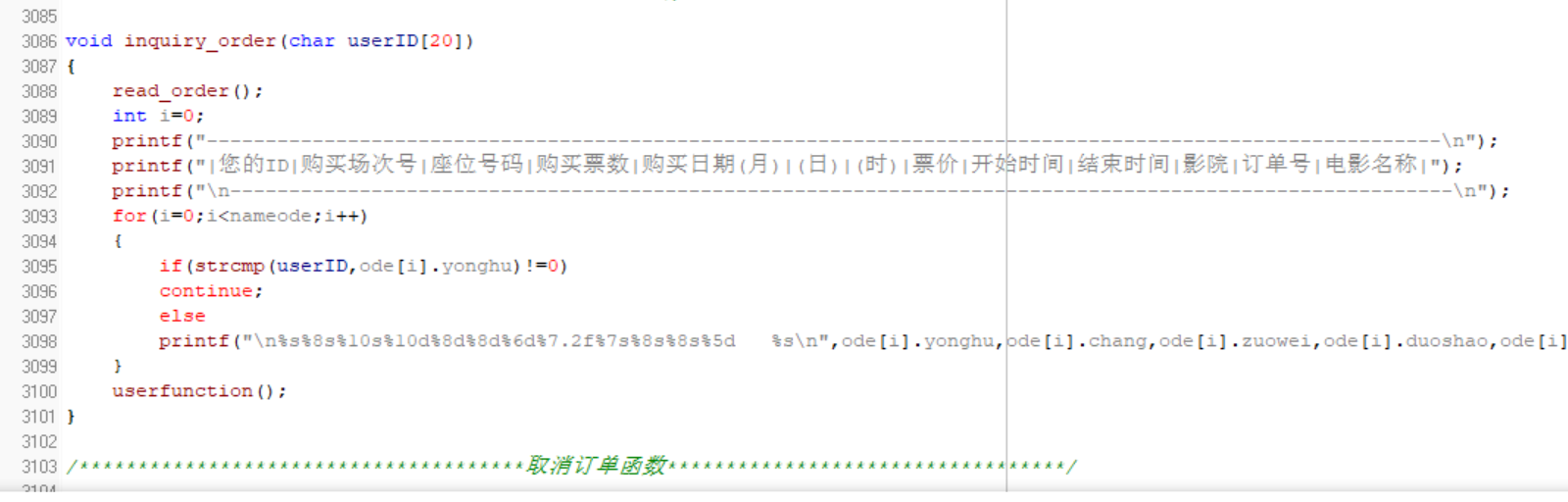
最后生成订单信息，写入文件中，并通过printf函数打印文件信息，展示给用户订单信息，完成购票。

**个人信息管理**：



将newmail，newpass，newphone录入字符串并覆盖原有存在字符串组中的admin[i].。此处用到了strcpy函数。

**查询订单：**



用了一个for函数，通过文件读取函数将所有的数据都进去结构体之中，在显示了该用户的所有订单之后，返回主菜单。

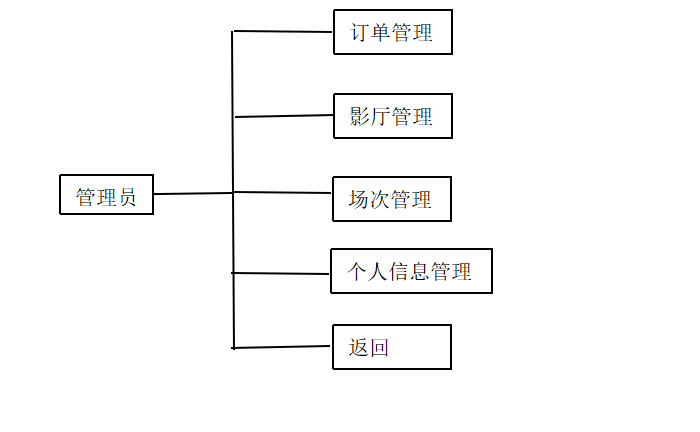
**取消订单：**

这里用了void cancel\_order()函数，让用户输入想要取消的订单号码

，取消订单之后又返回设置。

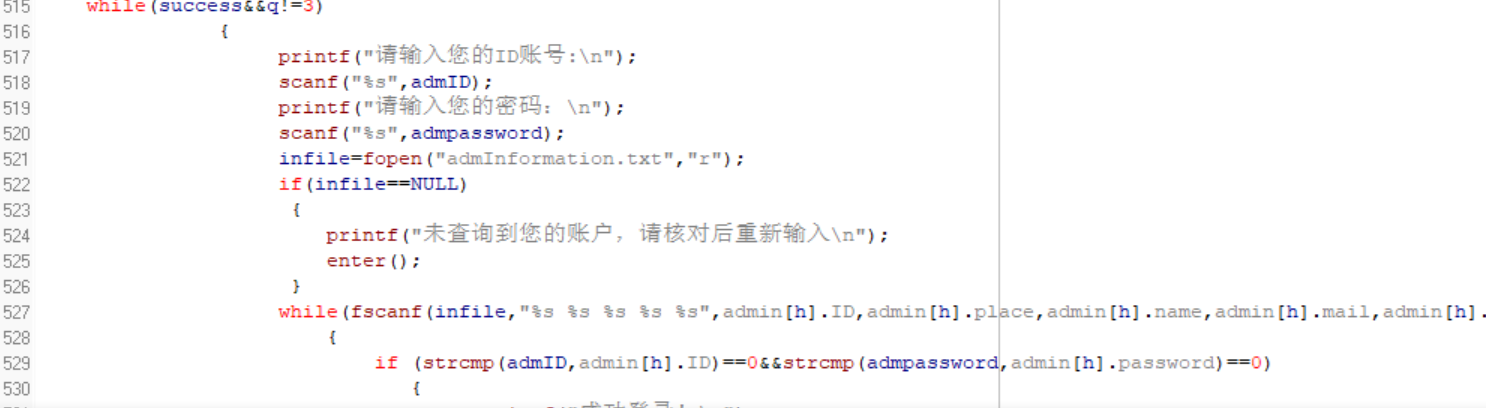


管理员部分：

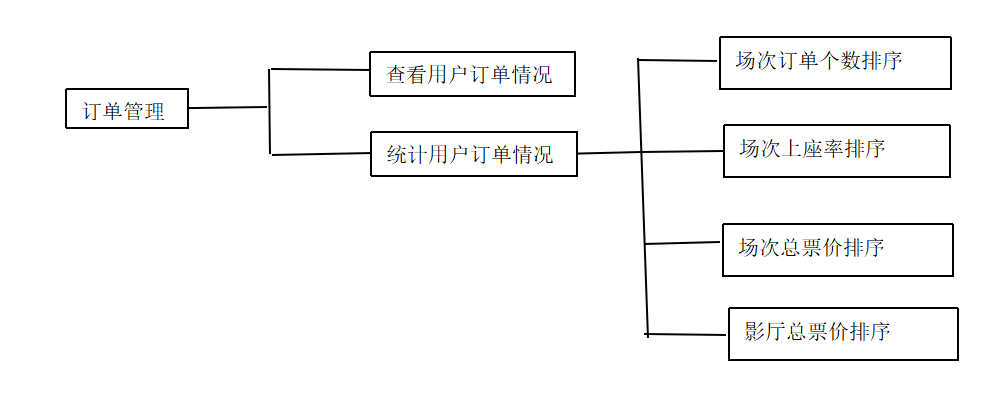


1. 登录页面

登录页面写成void admdenglu()函数



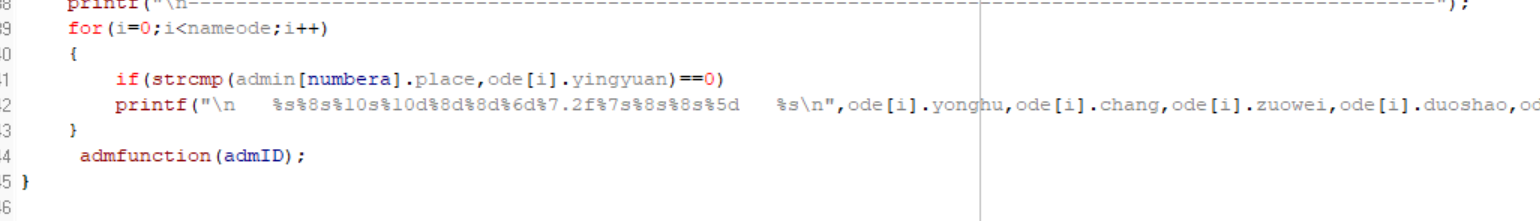
用while循环，检测输入账户ID是否存在在管理员文件信息中，如果不存在就给出提示，重新输入用户ID，在while循环中插入while循环，打开文件中的数据信息，如果ID和密码都匹配，就说明用户登录成功，如果不成功就重新输入用户ID和密码，只有三次输入密码的机会，如果三次都不成功就跳回主页面重新选择。



检验用户输入的数字是否为有效数字，用了if else函数，如果输入的数字不对或者输入的不是数字，显示重新输入。



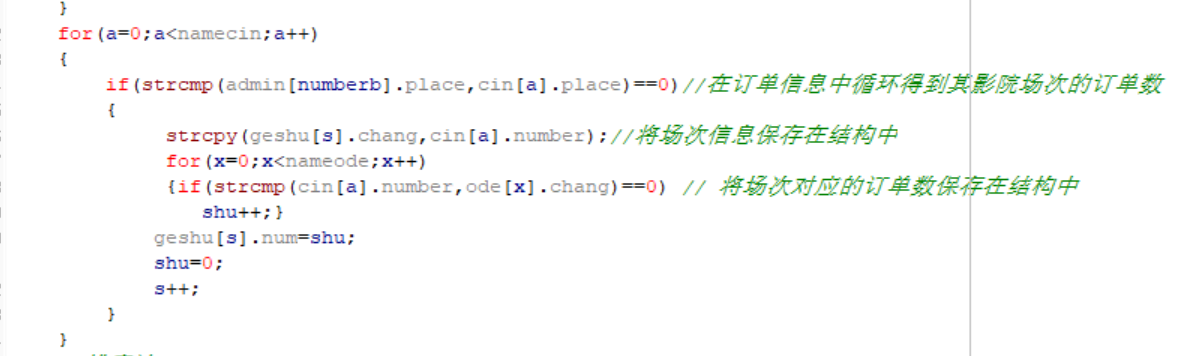
查看用户订单情况用了examine\_order(char admID[20])函数



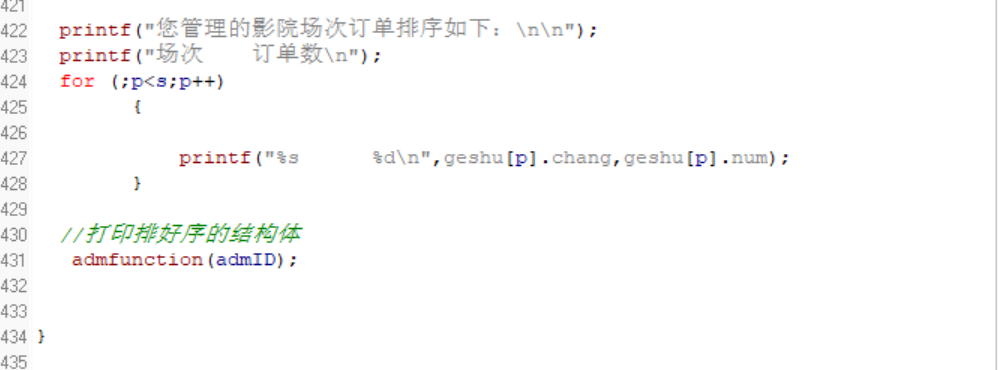
在 这个函数中，我们用了for循环去寻找符合条件的订单号，然后将它打印出来这样就可以极大程度的方便用户去读取订单，看清订单的各项信息。

统计用户订单情况分为四个部分，按照场次订单个数，场次上座率，场次总票价，影厅总票价四个方面来排序，

当按场次订单个数排序的时候，用了for和if循环，将对应的订单数保留在了结构中



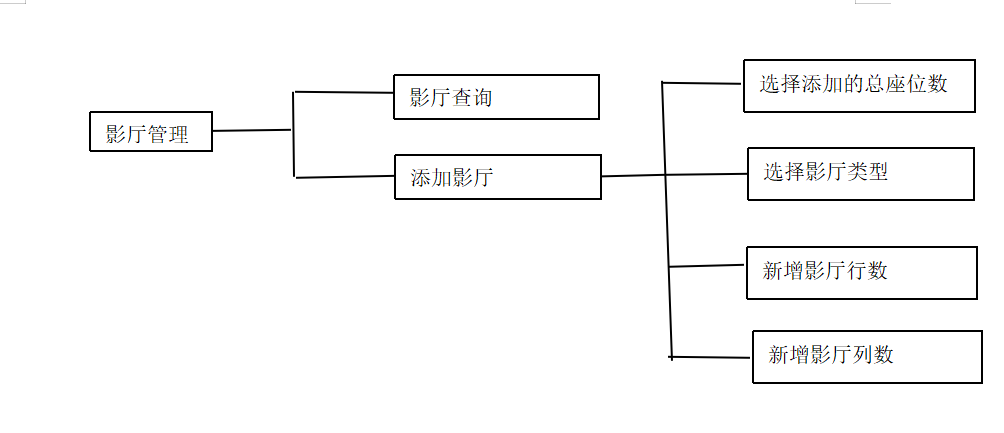
在进入影厅管理页面的时候，有两个选项



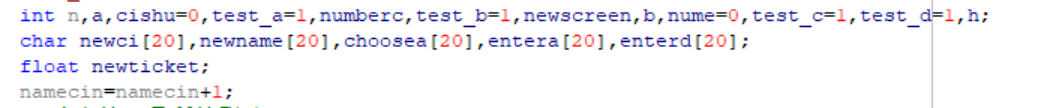
用了for循环打印了结构体里面的信息，生成了排序信息。px\_changci(char admID[20])

按照场次上座率排序的时候，整体设计思路和上面个函数相同，用了一个新的结构体，把两个信息连接起来，从而能够更加方便排序。px\_rate(char admID[20])用了这个函数。px\_yingting(char admID[20])排序影厅

按照场次总票价，影厅总票价排序的时候，整体设计思路都跟上面设计思路相同。



在影厅管理中，分为影厅查询和添加影厅，increase\_ci(char admID[20])函数来添加影厅

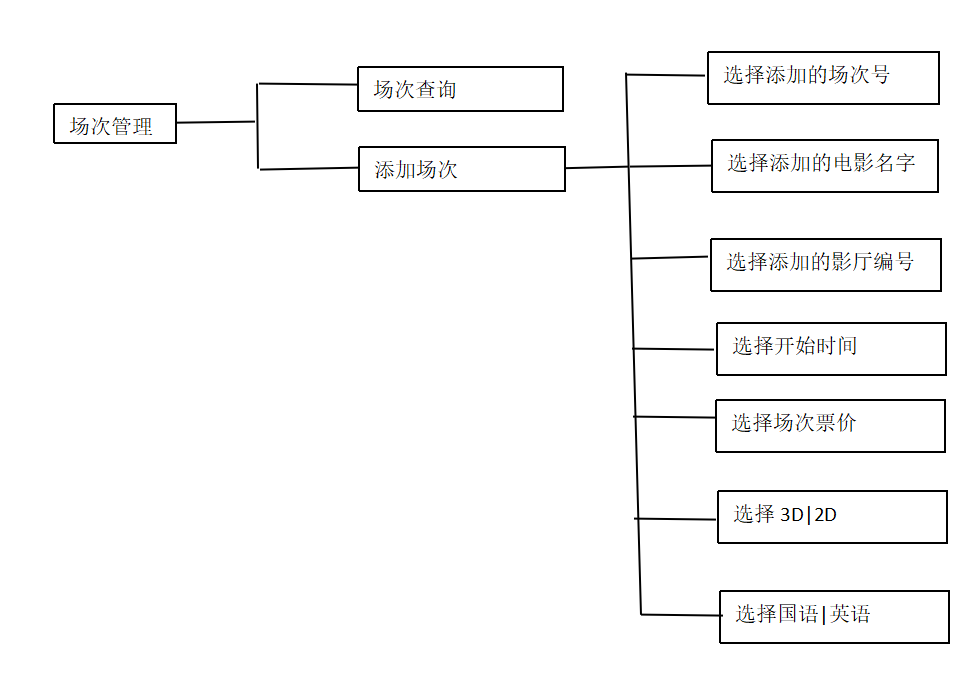


首先对这些变量进行声明

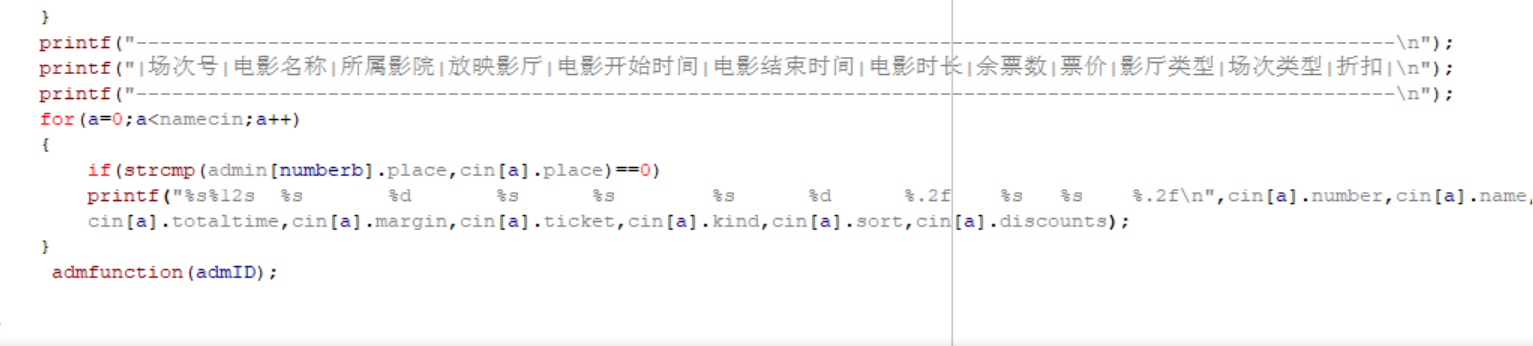


用dowhile循环里面插入ifelse循环，层层检验。

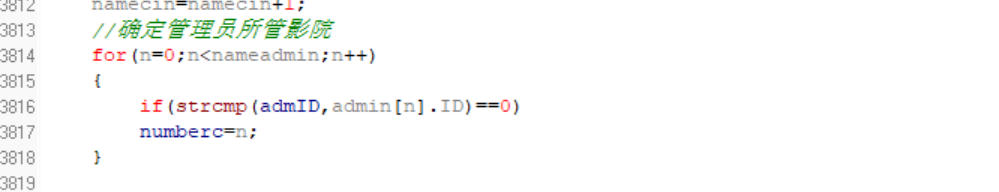
在场次管理页面的时候，分为场次查询和添加场次两个函数



在场次查询的时候，用了examine\_chang(char admID[20])这个函数，这个函数里面有一个传参，这个传参是一个字符型的数，用这个函数先去对比同一个影院的不同场次，结果出来方便管理员查看



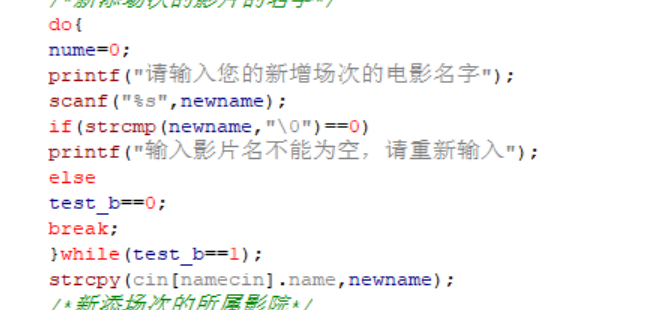
在添加场次的时候，用了void increase\_ci(char admID[20])函数，对添加的场次信息依次进行了检验



用for循环确定了管理员所管的影院



用dowhile循环确定了输入的场次号是否与这个管理员管理的影院中的其他场次名字重合，如果重合，重新输入。

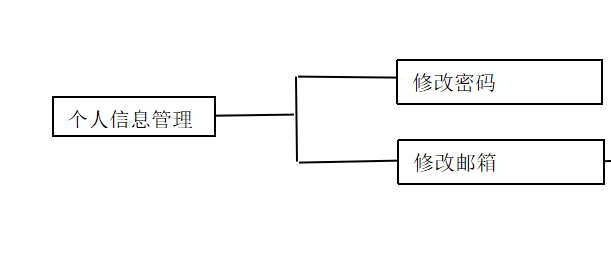


检验新添影片的名字是否为空，如果为空，重新输入。



在dowhile循环里面套ifelse函数，求出是否有同一个影院同一个影厅里面发生时间冲突的情况，如果发生了冲突显示和哪个场次冲突，然后让管理员重新输入。

修改个人信息



将newmail，newpass录入字符串并覆盖原有存在字符串组中的admin[i].password。此处用到了strcpy函数