# 2总体设计

1. 功能划分

该系统可按功能进行模块划分，如图所示

**图书管理**

**观看制作信息**

**账户注册**

**账户登入**

**图书管理系统**

**删**

**增**

**查**

**改**

**借还**

**浏览**

**增**

**删**

**用户**

**管理**

**查**

**改**

**浏览**

**黑名单管理**

**删**

**增**

**浏览**

1. 数据结构

本图书管理系统采用单向链表作为基础数据结构，如下是数据类型的定义:

图书包括书名，作者，价格用字符串所以在输入时需要判断字符是否符合价格的样式，当然不能完全考虑会有些bug，借书者身份证这个是来与借书者联系，方便管理图书，使书人同步联系，书的编号，是为了在借书者处联系借书者旗下的书籍，当然借书数量是要有上限的，无上限的话还没想过如何实现，也许可以再次使用单向链表嵌套，因为太麻烦，能力有限并且时间有限所以不做考虑，直接开了数组来作为借书上限。

struct books{ //图书的信息结构体

char book\_name[20]; //书名

char author\_name[20]; //作者

char price[10]; //价格

long long borrow; //借书者身份证

int id; //书的编号

struct books \*next;

}\*head\_book=NULL;

成员包括，账户，密码，姓名，电话号码，身份证号码，性别，用户级别，所借图书数量，所借图书。账户，电话号码及身份证号码应具有唯一性，所以在注册时需要遍历数据库内容，确保没有相同数据，身份证号码应具有18位判断下字符串长度（除以10的18次幂判断是否为0），及字符串不能出现除数字外的数，但是我并没有实现，因为身份证号码不仅仅与长度有关，还要判断真实性，所以就没有判断，其实就是不会，电话号码同理8或11位，也没有实现。性别采用整型变量存储，0代表女1代表男，其实是非0都是男，用户级别同性别方式存储。

struct members{

char account[15]; //账号

char password[15]; //密码

char name[15]; //姓名

long long phone,id; //电话号码+身份证号

int sex; //性别

int level; //用户级别

int cnt; //所借图书数量

int borrow[5]; //所借图书

struct members \*next;

}\*head\_member=NULL;

黑名单仅包含身份证id，也比较好实现，只不过拉黑要注意该账户是否存在，不存在直接拉黑，如果存在图书是否归还完全，归还完全才能拉黑，注册账户时要同样遍历数据库查看身份证是否被拉黑。

struct blacklist{

long long id;

struct blacklist \*next;

}\*head\_black=NULL;

下面是各个链表的生成，只需传入数据和头指针地址，因为数据是当开启应用时直接传入，所以头指针采用全局变量，该做法好处是便于调用，当我的函数嵌套较多时不用一直传参

//用户链表建立及增加

struct members \*creat\_members(struct members \*head,struct members data){

struct members \*p;

p=(struct members\*)malloc(sizeof(struct members));

data.next=NULL;

\*p=data;

if(head!=NULL)p->next=head;

return p;

}

//图书链表建立及增加

struct books \*creat\_books(struct books \*head,struct books data){

struct books \*p;

p=(struct books\*)malloc(sizeof(struct books));

data.next=NULL;

\*p=data;

if(head!=NULL)p->next=head;

return p;

}

//黑名单链表建立及增加

struct blacklist \*creat\_blacklist(struct blacklist \*head,struct blacklist data){

struct blacklist \*p;

p=(struct blacklist\*)malloc(sizeof(struct blacklist));

data.next=NULL;

\*p=data;

if(head!=NULL)p->next=head;

return p;

}

1. 程序流程

该系统可按功能进行模块划分，如图

**账号登入**

**账号注册**

**浏览制作人名单**

**管理者界面**

**主界面**

**会员用户界面**

**普通用户界面**

**图书信息浏览**

**图书信息查询**

**注销账户**

**修改本人信息**

**查询本人信息**

**借书**

**查询本人借阅情况**

**浏览图书信息**

**拉白身份证**

**注销用户信息**

**查询图书信息**

**添加图书信息**

**修改图书信息**

**删除图书信息**

**浏览用户信息**

**查询用户信息**

**用户级别修改**

**修改用户信息**

**还书**

**拉黑身份证**

**浏览黑名单**

3详细设计

**下为总体需要用到的函数和数据结构：**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下为基础数据管理\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct books{

char book\_name[20]; //书名

char author\_name[20]; //作者

char price[10]; //价格

long long borrow; //借书者身份证

int id; //书的ID

struct books \*next;

}\*head\_book=NULL;

struct members{

char account[15]; //账号

char password[15]; //密码

char name[15]; //姓名

long long phone,id; //电话号码+身份证号

int sex; //性别

int rank; //用户级别

int cnt; //所借图书数量

int borrow[5]; //所借图书

struct members \*next;

}\*head\_member=NULL;

struct blacklist{

long long id;

struct blacklist \*next;

}\*head\_black=NULL;

struct members \*creat\_members(struct members\*,struct members); //用户链表建立及增加

struct books \*creat\_books(struct books\*,struct books); //图书链表建立及增加

struct blacklist \*creat\_blacklist(struct blacklist\*,struct blacklist); //黑名单链表建立及增加

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下为主要界面\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void staff\_list(); //制作人员清单

void interface\_login(); //登入界面

void Manager\_interface(); //管理者界面

void interface\_rank0(struct members \*); //普通用户界面

void interface\_rank1(struct members \*); //会员用户界面

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下为用户功能\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void login(); //账号登入

void regis(); //账号注册

void browse\_members(); //浏览成员信息及管理

void find\_member(); //查找成员信息及管理

void modific\_member(); //管理员修改成员信息

void modific\_mymessage(struct members \*); //用户修改自己信息

void find\_mymessage(struct members \*); //用户查看自己信息及修改

void find\_mybook(struct members \*); //查询本人借阅书籍

void del\_member(long long); //删除成员信息

void print\_members(); //输出成员信息

void change\_rank(); //用户级别修改

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下为黑名单功能\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void add\_blacklist(); //加入黑名单

void browse\_blacklist(); //浏览黑名单

void del\_blacklist(); //拉白

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*下为图书管理功能\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void browse\_manage\_books(); //浏览图书信息及管理

void find\_manage\_book(); //查询图书信息及管理

void browse\_books(struct members \*); //用户浏览图书信息

void find\_book(struct members \*); //用户查询图书信息

void print\_books(); //输出图书信息

void add\_book(); //管理员添加图书信息

void del\_book(); //管理员删除图书信息

void modific\_book(); //管理员修改图书信息

void borrow\_book(struct members \*); //用户借书

void Back\_book();

由于函数过多，部分函数在下面我的程序里展示，这里将讲述比较重要的函数，迫于时间及能力原因，如明明也有力所能及的优化方案却没有实现，或者考虑不周，打开文件的操作我都是默认成功没有判断是否成功及后续操作，本想实现几个排序，发现有些麻烦时间紧迫先写了比较重要的功能，代码可能有许多地方做的不足，甚至出现较大bug，希望多多包含，欢迎指正。

**void interface\_login();**

首先自然是void interface\_login(); 这个函数是运行程序第一个函数，会将在.exe文件下读取

blacklist.txt，book.txt，user.txt，内的数据，如果缺少某个文件那么会生成该文件一个空文件，将读取的内容存储到链表中等待使用，然后该函数会展示接下来需要进行的操作。下为代码：

//主界面

void interface\_login(){

int n=1;

//读入用户数据

struct members m\_data;

FILE \*fp1;

fp1=fopen("user.txt","a+");

while(fscanf(fp1,"%s%s%s%lld%lld%d%d%d\n",m\_data.account,m\_data.password,m\_data.name,&m\_data.phone,&m\_data.id,&m\_data.sex,&m\_data.level,&m\_data.cnt)!=EOF){

int i;

for(i=0;i<5;i++)m\_data.borrow[i]=0;

for(i=0;i<m\_data.cnt;i++)

fscanf(fp1,"%d",&m\_data.borrow[i]);

head\_member=creat\_members(head\_member,m\_data);

}

fclose(fp1);

//读入黑名单

struct blacklist black\_data;

fp1=fopen("blacklist.txt","a+");

while(fscanf(fp1,"%lld\n",&black\_data.id)!=EOF){

head\_black=creat\_blacklist(head\_black,black\_data);

}

fclose(fp1);

//读入图书数据

struct books b\_data;

fp1=fopen("book.txt","a+");

while(fscanf(fp1,"%s%s%s%lld%d\n",b\_data.book\_name,b\_data.author\_name,b\_data.price,&b\_data.borrow,&b\_data.id)!=EOF){

head\_book=creat\_books(head\_book,b\_data);

}

fclose(fp1);

while(n){

system("cls");

printf("-------------欢迎进入图书管理系统 o(\*≧▽≦)ツ-------------\n");

printf("-----------------------------------------------------------\n");

printf("-----------------------------------------------------------\n");

printf("------------- 输入1 账号登入 -------------\n");

printf("------------- 输入2 账号注册 -------------\n");

printf("------------- 输入3 观看制作信息 -------------\n");

printf("------------- 输入0 关闭系统 -------------\n");

printf("-----------------------------------------------------------\n");

printf("-----------------------------------------------------------\n");

scanf("%d",&n);

switch(n){

case 1:login();break;

case 2:regis();break;

case 3:staff\_list();break;

}

}

}

**void borrow\_book(struct members \*);**

图书馆自然是要借书的，接下来将讲述void borrow\_book(struct members \*); 的实现，

用户登入用户界面时已经将自己信息的参数传入函数，借书时只需将借书者的信息地址做参数再传入到该函数内，这样以便调用更加快捷，先判断借书是否达到上限，如果达到那么无法借阅，再请用户输入目标图书编号，先查找图书是否存在，再判断图书是否被借出，如果都没有，那么成功借书，这时候需要打开存储文件修改文件内容和内存中链表的数据，图书默认借阅者longlong内数据存储为0即没有被借阅，只需修改这个，而读者需要修改借阅数量，以及借阅书籍的书籍编号。代码如下：

void borrow\_book(struct members \*user){

int a\_id;

struct books \*q\_book=head\_book;

if(user->cnt>4){

printf("用户所借书数量已达上限\n输入任意键返回……\n");

getch();

return;

}

printf("请输入目标图书的编号\n");

scanf("%d",&a\_id);

int flag=1;

if(q\_book==NULL){

printf("该书不存在\n输入任意键返回……\n");

getch();

return;

}

if(q\_book!=NULL){

while(q\_book->id!=a\_id&&q\_book->next!=NULL)q\_book=q\_book->next;

if(q\_book->id==a\_id)flag=0;

}

if(flag){

printf("该书不存在\n输入任意键返回……\n");

getch();

return;

}

if(q\_book->borrow){

printf("该书已被借出\n输入任意键返回……\n");

getch();

return;

}

user->borrow[user->cnt]=q\_book->id;

user->cnt++;

q\_book->borrow=user->id;

FILE \*fp1=NULL;

//修改用户数据

user=head\_member;

fp1=fopen("user.txt","w");

while(user!=NULL){

int i;

fprintf(fp1,"%s %s %s %lld %lld %d %d %d",user->account,user->password,user->name,user->phone,user->id,user->sex,user->level,user->cnt);

for(i=0;i<user->cnt;i++)fprintf(fp1," %d",user->borrow[i]);

fprintf(fp1,"\n");

user=user->next;

}

fclose(fp1);

//修改图书数据

q\_book=head\_book;

fp1=fopen("book.txt","w");

q\_book=head\_book;

while(q\_book!=NULL){

fprintf(fp1,"%s %s %s %lld %d\n",q\_book->book\_name,q\_book->author\_name,q\_book->price,q\_book->borrow,q\_book->id);

q\_book=q\_book->next;

}

fclose(fp1);

printf("借阅成功\n输入任意键返回……\n");

getch();

return;

}

**void Back\_book();**

有借自有还，还书函数void Back\_book();与借书类似修改信息，所以就不贴详细代码。

**void find\_mymessage(struct members \*);**

接下来讲 用户查看自己信息及修改void find\_mymessage(struct members \*);的实现，在用户界面直接将用户数据地址传入，然后根据数据输出，性别和会员用户根据01来分别进行不同输出，而图书比较麻烦，如果没借即q\_member->cnt为0直接输入暂无借阅，否则需要遍历q\_member->cnt次，当然有遍历一次的O(n)写法但是代码有点复杂，所以并没有这么写，而且借书最多5本，5O(n)线性复杂度应该也不会太慢，至于O(1)写法可能需要我大改数据存储方式使用嵌套存储，但是时间紧迫写到这时不可能大改，或者还有能基于我这存储方式的O(1)写法只是我没有想到。下为代码：

void find\_mymessage(struct members \*q\_member){

int n=1;

while(n){

system("cls");

printf("姓名 %s\n",q\_member->name);

printf("身份证号码 %lld\*\*\*\*\*\*\n",q\_member->id/1000000);

printf("账号 %s\n",q\_member->account);

printf("密码 %s\n",q\_member->password);

printf("电话号码 %lld\n",q\_member->phone);

if(q\_member->sex)printf("性别 男\n");

else printf("性别 女\n");

if(q\_member->level)printf("会员用户\n");

else printf("普通用户\n");

if(q\_member->cnt)printf("所借图书如下\n");

else printf("暂无借阅图书\n");

int i=0;

for(i=0;i<q\_member->cnt;i++){

printf("所借图书如下\n");

printf("书籍编号 书名 作者 价格\n");

int i;

for(i=0;i<q\_member->cnt;i++){

struct books \*q\_book=head\_book;

while(q\_book->id!=q\_member->borrow[i]&&q\_book->next!=NULL)

q\_book=q\_book->next;

printf("%-11d%-20s%-20s%-10s\n",q\_book->id,q\_book->book\_name,q\_book->author\_name,q\_book->price);

}

}

printf("-----------------------------------------------------------\n");

printf("-----------------------------------------------------------\n");

printf("------------- 输入1 修改用户信息 -------------\n");

printf("------------- 输入0 返回上一级 -------------\n");

printf("-----------------------------------------------------------\n");

printf("-----------------------------------------------------------\n");

scanf("%d",&n);

switch(n){

case 1:modific\_mymessage(q\_member);break;

}

}

}