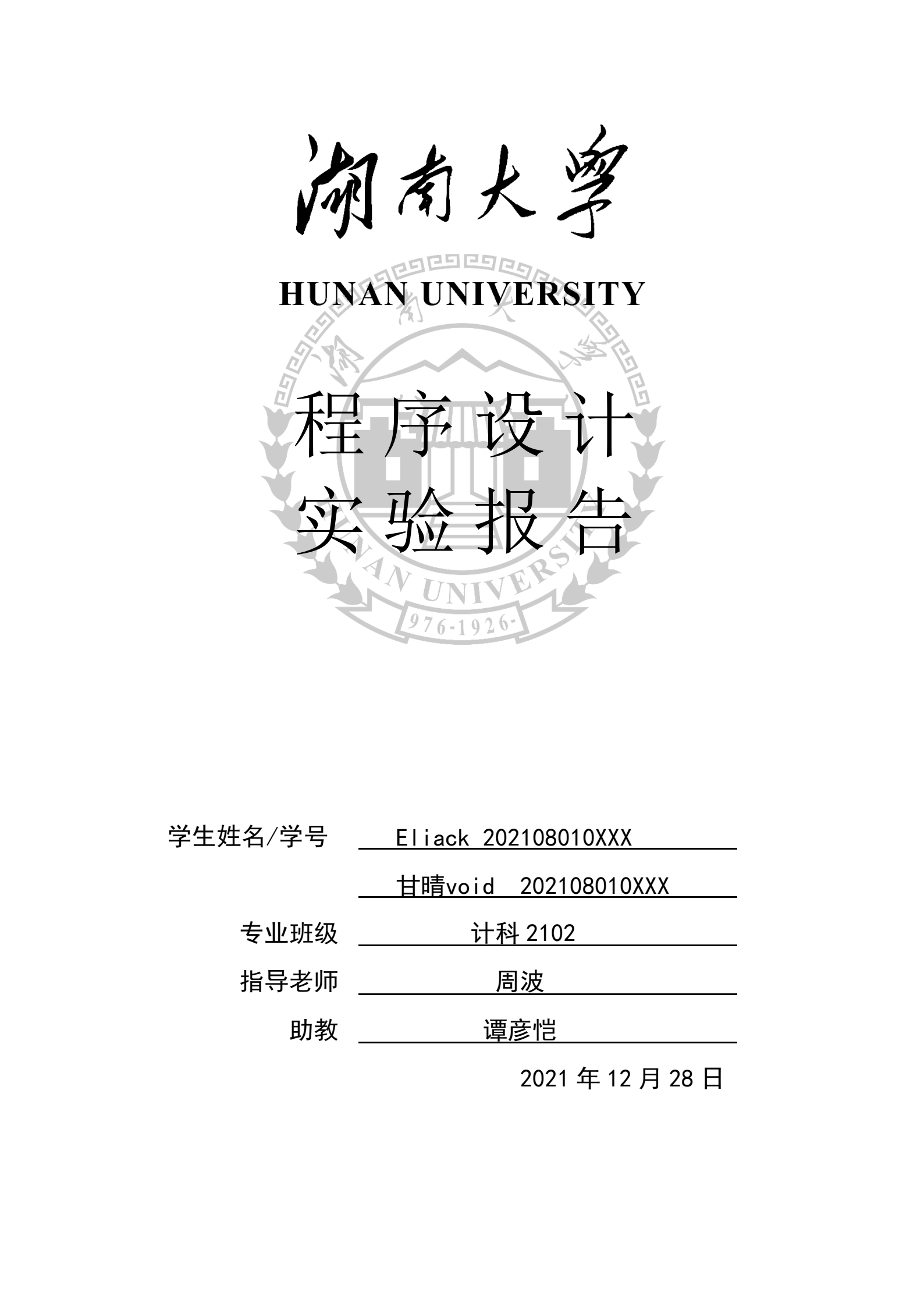

湖南大學

HUNAN UNIVERSITY

程序设计
实验报告

学生姓名/学号 Eliack 202108010XXX

甘晴void 202108010XXX

专业班级 计科 2102

指导老师 周波

助教 谭彦恺

2021 年 12 月 28 日

目录

图书管理系统

1、 问题描述

H 大学图书馆邀请你建立一个图书馆信息管理系统。请使用面向对象思想完成该问题，具体要求如下：

一、设计一款文字式交互的图书管理系统；
二、图书馆必须支持至少 10000 册书存储，如果可实现书籍可动态增长，加分
三、图书信息包含：

- 题名
- ISBN/ISSN
- 作者
- 分类号（分类规则自定，要求有三级分类，可参考中图分类法）

四、图书馆系统提供两种用户模式，请为他们设计不同的用户类：
1）管理员模式：

- 系统最初提供一个默认的管理员账户以及默认密码；
- 管理员具备以下功能：
  - 可以使用管理员账号登录
  - 支持对学校用户的账号进行基本管理，添加、删除学校用户默认账号和密码，默认账号为学号/教师编号，密码为 123456；恢复学校用户默认密码；
  - 管理员可以对图书信息进行修改
  - 管理员可以增加、删除、搜索图书

2）学校用户模式（学校用户超过 5 千人）：

- 学校用户可以通过账号和密码登录，账号为学号/教师编号，密码为 123456;
- 学校用户可以修改自己的密码
- 学校用户可以搜索图书
- 学校用户可以借、还图书
- 学校用户可以查看自己的借阅记录

五、设计图书馆类，包含馆藏图书列表、用户列表等成员、在馆记录、用户借阅记录等。

六、图书馆系统提供根据任一信息的搜索图书功能：

- 题名，精确查找到书
- ISBN/ISSN，精确查找到书
- 作者，模糊查找到该作者所有书，字典序排序
- 分类号，三级分类，每一级分类均可模糊查找到书，字典序排序，按页显示；如，N 自然科学总论——TP 自动化技术、计算机技术——TP3 计算机技术。
  在以上每一级时，均会出现该级所有数目，字典排序，按页显示；不明白意思的同学，可以自行登入学校图书馆系统，进入搜索书目功能中，点击分类导航选项，然后进入每级分类的页面看看

- 搜索不考察性能，仅考察功能

七、加分项（总分不超过 100 分）
（1）支持大数据，比如书籍记录突破百万，用户数量突破万级规模；
（2）贴近实际的图书馆管理系统，新增若干功能等；
（3）实现文件的创建、读、写等操作；
（4）考虑用户体验，如使用方便度等；
（5）搜索时性能考察，调查、思考、设计加强搜索性能的方式，此项仅适合学有余力的同学；

2、 分析设计

思路分析及原理

1. 要建立一个图书馆信息管理系统。自然有图书 book 类，用户 user 类，管理员 administrator 类。为了将他们统一，还得有一个图书馆 library 大类
2. 图书馆要实现两种用户模式和一系列功能。开始时需要有一个菜单供使用者选择，显示这个菜单的功能作为 library 类中的一个成员函数。选择不同的模式则通过 switch，case 的结构来实现。
3. 当选择某种模式时，会出现账号密码登录的界面，代码直接写在 main 函数中。
4. 登陆后要提示使用者该种模式的各种功能。选择不同的功能同样使用 switch，case 结构。
5. 当修改数据结构中的某个数据时或者添加某个新数据时，需要将数据保存到 txt 中，因此要用到读文件和写文件，为了不混淆各种类的读文件和写文件，则每个类都有读文件和写文件函数
6. 当执行功能时，融合清屏，暂停等操作实现界面的整洁；通过 goto 可以随时返回到执行的上一页。

数据结构的定义

均定义在 library 的大类中

1. book booksarray[100000]　book 类的数组代表 100000 储存量的图书（规模可达到 10 万及以上）
2. user userarray[8000]　user 类的数组代表 8000 储存量的用户
3. administrator adminarray[100] administrator 类的数组代表 100 储存量的管理员
4. record recordarray[100000] record 类的数组代表 100000 储存量的图书借阅记录

函数设计、参数规格以及返回类型：

1.全局变量部分

在 main 函数所在 cpp 中，main 函数外定义了一个图书馆 library 类的全局变量 hnu.。

2.类

1. book 类
   属性：题目 string topic、ISBN/ISSN string type、作者 string writer、分类号 string category、借出状态 int state、借阅者账号 long ownerid。

2. user 类
   属性：账号 long account、密码 string password。

   成员函数：void usermenu() 显示用户功能菜单

3. administrator 类
   属性：账号 long account、密码 string password。
   成员函数：void administratormenu() 显示管理员功能菜单
   void manage_accountmenu() 显示管理用户系列功能菜单
   void modify_bookmenu() 显示管理图书系列功能菜单
   void book_management_menu();显示修改图书信息界面菜单

4. record 类
   属性：账号 long ownerid、题目 string topic、作者 string writer、借阅状态 int state.。

5. library 类
   属性：书的数量 int booksnum、借阅记录的数量 int recordnum、用户的数量 int usernum、管理员的数量 int adminnum、书类数组 book booksarray[10000]、用户类数组 user userarray[8000]、管理员类数组 administrator adminarray[100]、借阅记录类数组 record recordarray[10000]。

   成员函数：(选择类内声明,类外定义分文件编写）
   //user.h
   library(); 构造函数
   void iniuser(); 从文件中读用户信息函数
   void save(); 从文件中写用户信息函数

   //library.cpp
   void signmenu(); 显示登录界面菜单
   void zongmenu(); 显示用户模式菜单
   void checkmenu(); 显示检索信息种类菜单
   void xinxitypemenu(); 显示检索信息种类菜单
   //void developerinfor()：显示开发者函数

   //books.h
   int getbooksnum(); 获取图书数量函数
   void inibooks(); 从文件中读图书信息函数
   void booksave(); 从文件中写图书信息函数
   void showbook(book Book); 展示图书信息函数
   void search(); 图书搜索函数

void borrowbook(long); 借图书函数
void givebook(long); 还图书函数

//borrowrecord.h
void userborrowrecord(long); 显示特点借阅者的借阅记录
void allborrowrecord(); 显示所有借阅记录函数
void recordsave(book); 从文件中写借阅记录函数
void inirecord(); 从文件中读借阅记录函数
int getrecordnum(); 获取借阅记录数量函数

//login_check.h
int login_user(int account,string password);
判定用户是否登陆成功，否则返回错误类型
int login_admin(int account,string password);
判定管理员是否登陆成功，否则返回错误类型

//admin.h
void admin_save(); 从文件中写管理员信息函数
void iniadmin(); 从文件中读管理员信息函数
void addaccount(); 添加新用户函数
void deleteaccount(); 删除某用户函数
void modifypassword(); 修改用户信息函数
void resetpassword(); 恢复默认账号密码函数
void addbook(); 添加图书函数
void deletebook(); 删除图书函数
void modifybook(); 修改图书信息函数

3、运行测试

一、登录页面

显示登录菜单

使用 library 类中的 signmenu()成员函数

```
void library::signmenu(){
cout<<"**************************************"<<endl;
cout<<"******欢迎来到湖南大学图书馆******"<<endl;
cout<<"**********请选择用户模式**********"<<endl;
cout<<"**************************************"<<endl;
cout<<"**************************************"<<endl;
cout<<"******      1.用户模式      ******"<<endl;
cout<<"******      2.管理员模式    ******"<<endl;
cout<<"******    0.查看开发者信息  ******"<<endl;
cout<<"**************************************"<<endl;
}
```

二、用户模式

显示用户功能菜单

使用 library 类中的 usermenu()成员函数

```
void user::usermenu(){
cout<<"*************************************"<<endl;
cout<<"***** 1.修改密码              *****"<<endl;
cout<<"***** 2.搜索图书              *****"<<endl;
cout<<"***** 3.借图书                *****"<<endl;
cout<<"***** 4.还图书                *****"<<endl;
cout<<"***** 5.查看借阅记录          *****"<<endl;
cout<<"***** 0.退出登录              *****"<<endl;
cout<<"*************************************"<<endl;
}
```

1.修改密码

直接在 main 函数中

进行修改

修改好后使用 usersave()函数进行保存

并且返回上一级用户模式功能目录

```
system("cls");
cout<<"请输入新密码:"<<endl;
cin>>hnu.userarray[temp_account].password;
cout<<"请确认你的新密码:"<<endl;
cin>>hnu.userarray[temp_account].password;
hnu.save();
cout<<"修改成功"<<endl;
system("pause");
system("cls");
goto userfunc;
break;
```

2.**搜索图书**

使用 library 中 search()成员函数进行搜索

其中要用到显示信息种类 xinxitype()函数提示搜索所用到的的信息、提醒输入不同的数字以代表不同的搜索信息。

搜索后返回用户功能界面。

1.题名搜索

```
if(xinxitype==1){
      cout<<"请输入题名"<<endl;
      string topic_;
      cin>>topic_;
      int i=0;
      for(;i<this->booksnum;i++){
      if(topic_==this->booksarray[i].topic){
      this->showbook(this->booksarray[i]);
      return;
      }
      }
      cout<<"找不到该书"<<endl;
      return;
      }
```

2. ISBN/ISNN 搜索

```
else if(xinxitype==2){
      cout<<"请输入ISBN/ISSN"<<endl;
      string type_;
      cin>>type_;
      int i=0;
      for(;i<=this->booksnum;i++){
      if(type_==this->booksarray[i].type){
      this->showbook(this->booksarray[i]);
      return;
      }
      }
      cout<<"找不到该书"<<endl;
      return;
      }
```

3.作者搜索

```
else if(xinxitype==3){
      cout<<"请输入作者名"<<endl;
      string writer_;
      cin>>writer_;
      int i=0;
      int sum1=0;
      for(;i<=this->booksnum;i++){
      if(writer_==this->booksarray[i].writer){
      sum1++;
      cout<<sum1<<".";
      this->showbook(this->booksarray[i]);
      }
      }
      if(sum1==0){
      cout<<"找不到该作者的书籍"<<endl;
      return;}else return;
      }
```

4.分类号搜索

这里要用到三级分类号搜索，由于分类号的格式是固定的且在 txt 中是一个字符串，所以需要提取各个位置的字符串以分离各级分类号，这里写了一个函数以分离各级分类号，在搜索中依次检索各级分类号，并且搜索到的书籍若大于 10，又融合清屏等操作模拟了翻页。

```
else if(xinxitype==4){//new
     string onecategory_,twocategory_,threecategory_;
     int a1,a2,a3;
     cout<<"请输入一级分类号"<<endl;
     cin>>onecategory_;
 onesearch:int sum1=0;
     for(int i=0;i<=this->booksnum;i++){

        if(onecategory_[0]==this->booksarray[i].category[0]){
        sum1++;
        cout<<sum1<<".";
        this->showbook(this->booksarray[i]);
        if(sum1%10==0){
        cout<<endl;
        cout<<"第"<<sum1/10+1<<"页已到底"<<endl;
        cout<<"1.继续下一级搜索"<<endl;
        cout<<"2.退出搜索"<<endl;
        cout<<"3.下一页"<<endl;
        int select;cin>>select;
        if(select==1){a1=1;system("pause");system("cls");goto search2;}
        if(select==2){a1=2;system("pause");system("cls");goto search2;}
        if(select==3){system("pause");system("cls");}
        }
        }}
        if(sum1==0){cout<<"没有找到书籍"<<endl;return;}
        cout<<"1.继续搜索"<<endl;
        cout<<"2.退出搜索"<<endl;
        cin>>a1;

search2:   if(a1==1){
           system("cls");
           cout<<"请输入二级分类号"<<endl;
           cin>>twocategory_;
      twosearch:int sum2=0;
```

```
            for(int i=0;i<=this->booksnum;i++){
              int end1=booksarray[i].category.find("/");
              string p=match(booksarray[i].category,1,end1);
              if(onecategory_[0]==this->booksarray[i].category[0]&&twocategory_==p){
              sum2++;
              cout<<sum2<<"."<<endl;
              this->showbook(this->booksarray[i]);
              if(sum2%10==0){
              cout<<endl;
              cout<<"第"<<sum1/10+1<<"页已到底"<<endl;
              cout<<"1.继续下一级搜索"<<endl;
              cout<<"2.返回上一级搜索"<<endl;
              cout<<"3.退出搜索"<<endl;
              cout<<"4.下一页"<<endl;
              int select;cin>>select;
              if(select==1){a2=1;system("pause");system("cls");goto search3;}
              if(select==2){a2=2;system("pause");system("cls");goto search3;}
              if(select==3){a2=3;system("pause");system("cls");goto search3;}
              if(select==4){system("pause");system("cls");}
              }
              }}
              if(sum2==0){cout<<"没有找到书籍"<<endl;return;}
              cout<<"1.继续搜索"<<endl;
              cout<<"2.返回上一级搜索"<<endl;
              cout<<"3.退出搜索"<<endl;
              cin>>a2;

search3:        if(a2==1){
                system("cls");
                cout<<"请输入三级分类号"<<endl;
                cin>>threecategory_;
            threesearch:int sum3=0;
                for(int i=0;i<=this->booksnum;i++){
                int begin=booksarray[i].category.find("/")+1;
              int end=booksarray[i].category.size();
              string p1=match(booksarray[i].category,1,begin-1);
              string p2=match(booksarray[i].category,begin,end);
              if(onecategory_[0]==this->booksarray[i].category[0]&&twocategory_==p1&&threecategory_==p2){
              sum3++;
              cout<<1<<".";
              this->showbook(this->booksarray[i]);
              }}
              if(sum3==0){cout<<"没有找到书籍"<<endl;return;}
              cout<<"1.返回上一级搜索"<<endl;
              cout<<"2.退出搜索"<<endl;
              cin>>a3;
                if(a3==1) goto twosearch;
                else if(a3==2) return;
                else{
                system("cls");
                goto threesearch;
                 }
              }else if(a2==2) goto onesearch;
              else if(a2==3) return;
              else{
              system("cls");
              goto twosearch;
              }
            }else if(a1==2){
            return;
            }else{
            system("cls");
            goto onesearch;
            }
  }else{
  goto searchfunc;
  }}
```

3.借图书

同样是四种信息来搜索图书从而借图书，而在借图书中通过四种信息来借图书都是类似的，这里只放一个代码。

当用户输入书的编号选择借书时，会判断书籍是否已经被借阅，如果已经借阅则会提示，借阅后返回用户功能界面。

```
void library::borrowbook(long id){
borrowfunc:  system("cls");
     cout<<"你要借出哪本图书？请搜索"<<endl;
     this->xinxitypemenu();
     cout<<"输入信息的种类"<<endl;
     int xinxitype;
     cin>>xinxitype;
if(xinxitype==1){
     cout<<"请输入题名"<<endl;
     string topic_;
     cin>>topic_;
     int i=0;
     for(;i<this->booksnum;i++){
     if(topic_==this->booksarray[i].topic){
      cout<<"搜索成功"<<endl;
      this->showbook(this->booksarray[i]);
       if(this->booksarray[i].state==1){
          cout<<"你确定要借该书吗？ 借书请输入1，否则输入0"<<endl;
          bool borrow;cin>>borrow;
          if(borrow){
          cout<<"你已成功借出"<<" 《"<<this->booksarray[i].topic<<"》 "<<endl;
          this->booksarray[i].state=0;
          this->booksarray[i].ownerid=id;
          this->booksave();
          this->recordsave(booksarray[i]);
          return;
          }else{
          cout<<"你已取消本次操作"<<endl;
          return;
          }
       }else if(this->booksarray[i].state==0){
       cout<<"已借出，你不可借阅"<<endl;
       return;
       }
     }}
     cout<<"找不到该书"<<endl;
     return;
     }
```

4.还图书

首先要显示该用户已经借阅的书籍，然后会提示归还书籍

还图书后会返回用户功能界面。

```
void library::givebook(long id){
borrowfunc:  cout<<"你已借阅的图书："<<endl;
     int sum=0;
     for(int i=0;i<this->booksnum;i++){
     if(this->booksarray[i].ownerid==id){
     sum++;
     cout<<sum<<"."<<"该书的编号："<<i<<endl;
     cout<<"  该书的题名："<<this->booksarray[i].topic<<endl;
     cout<<"  该书的ISBN/ISNN："<<this->booksarray[i].type<<endl;
     cout<<"  该书的作者："<<this->booksarray[i].writer<<endl;
     cout<<"  该书的分类号："<<this->booksarray[i].category<<endl;
     }}
     if(sum==0){
     cout<<"你还未借阅书籍"<<endl;
     }else{
     cout<<"请输入你所要还的书的编号:"<<endl;
     int bookid;cin>>bookid;
     cout<<"你已成功归还"<<"《"<<this->booksarray[bookid].topic<<"》"<<endl;
     this->booksarray[bookid].state=1;
     this->booksarray[bookid].ownerid=0;
     this->booksave();
     sum--;
     if(sum>0){
     cout<<"请问你是否需要继续归还？1-是 2-否"<<endl;
     int select;cin>>select;
     if(select==1)
     goto borrowfunc;
     else return;}
     else return;
     }
}
```

5.查看借阅记录

显示某个用户的借阅记录

```
void library::userborrowrecord(long id){//new
     int sum=0;
     for(int i=recordnum-1;i>=0;i--){
        if(recordarray[i].ownerid==id){
        sum++;
        cout<<sum<<".";
        cout<<recordarray[i].topic<<" ";
        cout<<recordarray[i].writer<<" ";
        for(int j=0;j<booksnum;j++){
         if(recordarray[i].topic==booksarray[j].topic){
         recordarray[i].state=booksarray[j].state;
           if(recordarray[i].state==1){
           cout<<"已归还"<<endl;
           break;
          }else{cout<<"未归还"<<endl;
          break;}
         }
        }
     }
     }
     if (sum==0) cout<<"您未借阅任何图书"<<endl<<"即将返回上级目录"<<endl;

     return;
}
```

查看完后会返回用户功能界面。

0. 退出登录

通过 goto 返回上一级

```
case 0://退出登录
        cout<<"是否确认退出？"<<endl;
        cout<<"0：取消    1：确认"<<endl;
        int temp4;
        cin>>temp4;
        if (temp4==1)
        {
            cout<<"退出登录成功，即将返回登录界面"<<endl;
            cout<<"欢迎下次使用"<<endl;
            system("pause");
            system("cls");
            goto sign;//返回登录界面
        }
        else
        {
            cout<<"已取消，即将返回上一级界面"<<endl;
            system("pause");
            system("cls");
            goto userfunc;//返回第二级目录
        }
        break;
default:
        goto userfunc;
        break;
        }
        }
```

三、管理员模式

显示用户功能菜单

使用 library 类中的 administratormenu()成员函数

```
void administrator::administratormenu(){
cout<<"*********************************"<<endl;
cout<<"***** 1.管理用户账号        *****"<<endl;
cout<<"***** 2.修改图书信息        *****"<<endl;
cout<<"***** 3.搜索图书信息        *****"<<endl;
cout<<"***** 4.查看借阅情况        *****"<<endl;
cout<<"***** 5.查看馆藏图书列表    *****"<<endl;
cout<<"***** 6.查看用户列表        *****"<<endl;
cout<<"***** 7.查看在馆记录        *****"<<endl;
cout<<"***** 0.退出登录            *****"<<endl;
cout<<"*********************************"<<endl;
}
```

1. 管理用户账号

```
void administrator::manage_accountmenu(){
cout<<"*********************************"<<endl;
cout<<"***** 1.添加用户账号        *****"<<endl;
cout<<"***** 2.删除用户账号        *****"<<endl;
cout<<"***** 3.重置用户密码        *****"<<endl;
cout<<"***** 4.初始化用户密码      *****"<<endl;
cout<<"***** 0.返回上一级目录      *****"<<endl;
cout<<"*********************************"<<endl;
}
```

1. 添加用户账号

在上级目录“管理用户账号”下选择该功能后，自动调用 addaccount 函数，实现后返回清屏并返回上级目录“管理用户账号”。

```
case 1:
    hnu.addaccount();
    system("pause");
    system("cls");
    goto accountmenufunc;//返回第二级目录
    break;
```

首先需要判断用户规模是否已达上限，达到上限后将不能添加；

然后有一个账号查重的问题，如果即将要添加的账户与已经存在的账户重复，需要提示并让管理员重新添加；

若都无问题，则提示管理员输入密码并保存进现行数组，并及时保存至 txt 文件内；

```
void library::addaccount()//管理员权限：：添加账号
{
    system("cls");
    usernum++;
    if (usernum>7999) //处理越界问题
    {
        cout<<"用户规模已经达到上限，请先删除部分用户"<<endl;
        return;
    }
retry:
    cout<<"请输入要添加的账号:"<<endl;
    cin>>userarray[usernum-1].account;
    for (int i=0;i<=usernum-2;i++)//处理添加账户重复问题
    {
        if (userarray[usernum-1].account==userarray[i].account)
        cout<<"要添加的账号重复，请重新输入"<<endl;
        system("pause");
        system("cls");
        goto  retry;
    }
    cout<<"请输入密码:"<<endl;
    cin>>userarray[usernum-1].password;
    ofstream ofs;
    ofs.open(FILENAME,ios::out|ios::app);
    ofs<<userarray[usernum-1].account;
    ofs<<" ";
    ofs<<userarray[usernum-1].password;
    ofs.close();
    cout<<"添加成功"<<endl;
    cout<<"即将返回上级目录"<<endl;
    return;
}
```

2. 删除用户账号

在上级目录“管理用户账号”下选择该功能后，自动调用 deleteaccount 函数，实现后返回清屏并返回上级目录“管理用户账号”。

```
case 2:
    hnu.deleteaccount();
    system("pause");
    system("cls");
    goto accountmenufunc;//返回第二级目录
    break;
```

首先需要判断该账户是否存在，如果该账户不存在，则输出“未找到该账户”，并返回上级目录；若该账户存在，则将该账户的 account 值赋为 0 表示该账户已经被删除。

```
void library::deleteaccount()//管理员权限：：删除账号
{
    long temp_account;
    system("cls");
    cout<<"请输入要删除的账号:"<<endl;
    cin>>temp_account;
    for (int i=0;i<=usernum-1; i++)
    {
        if (userarray[i].account==temp_account)
        {
            userarray[i].account=0;
            this->save();
            cout<<"删除成功"<<endl;
            cout<<"即将返回上级目录"<<endl;
            return;
        }
    }
    cout<<"未找到该账号"<<endl;
    return;
}
```

3. 重置用户密码

在上级目录“管理用户账号”下选择该功能后，自动调用 modifypassword 函数，实现后返回清屏并返回上级目录“管理用户账号”。

```
case 3:
    hnu.modifypassword();
    system("pause");
    system("cls");
    goto accountmenufunc;//返回第二级目录
    break;
```

首先搜索需要重置的账号，若账户不存在，则输出“未找到该账号”并返回；

找到该账户后需要两次读入新密码，若两次读入的新密码不一致，则修改失败，需要重新进行该操作，若两次读入的密码一致则表示修改成功，先存储该密码，并保存至本地 txt，然后返回上级目录

```
void library::modifypassword()//管理员权限：：修改用户密码
{
    long temp_account;
    string temp_password,temp2_password;
    system("cls");
    cout<<"请输入要重置的账号:"<<endl;
    cin>>temp_account;
repasswordfunc:
    cout<<"请输入要重置的密码:"<<endl;
    cin>> temp_password;
    cout<<"请确认要重置的密码:"<<endl;
    cin>> temp2_password;
    if (temp_password!=temp2_password)
    {
        cout<<"密码不一致，请重新输入密码"<<endl;
        system("pause");
        system("cls");
        goto repasswordfunc;
    }
    for (int i=0;i<=usernum-1; i++)
    {
        if (userarray[i].account==temp_account)
        {
            userarray[i].password=temp_password;
            this->save();
            cout<<"修改成功"<<endl;
            cout<<"即将返回上级目录"<<endl;
            system("pause");
            return;
        }
    }
    cout<<"未找到该账号"<<endl;
    return;
}
```

4. 初始化用户密码

在上级目录“管理用户账号”下选择该功能后，自动调用 resetpassword 函数，实现后返回清屏并返回上级目录“管理用户账号”。

```
case 4:
    hnu.resetpassword();
    system("pause");
    system("cls");
    goto accountmenufunc;//返回第二级目录
    break;
```

初始化用户密码的作用是将所有的用户密码全部重置为初始密码 123456，若需要局部初始密码，在修改账户密码界面单独修改即可。

步骤比较简单，容易理解。

```
void library::resetpassword()//管理员权限：：初始化用户密码
{
  ifstream ifs;
  ifs.open(FILENAME,ios::in);
  for(int i=0;i<usernum;i++){
  userarray[i].password="123456";
  }
  this->save();
  ifs.close();
  cout<<"初始化所有用户密码成功"<<endl;
  cout<<"即将返回上级目录"<<endl;
    return;
}
```

0. 返回上级目录

在“管理用户账号”下选择该功能后，自动返回上级目录“管理员模式主菜单界面”。

```
case 0:
    cout<<"即将返回上级目录"<<endl;
    system("pause");
    system("cls");
    goto administrator_menu_func;//返回第一级目录
    break;
```

2. 修改图书信息

```
void administrator::book_management_menu(){
cout<<"*********************************"<<endl;
cout<<"***** 1.添加图书信息        *****"<<endl;
cout<<"***** 2.删除图书信息        *****"<<endl;
cout<<"***** 3.修改图书信息        *****"<<endl;
cout<<"***** 0.返回上一级目录      *****"<<endl;
cout<<"*********************************"<<endl;
}
```

1. 添加图书信息

在上级目录“修改图书信息”下选择该功能后，自动调用 addbook 函数，实现后返回清屏并返回上级目录“修改图书信息”。

```
case 1:
    hnu.addbook();
    system("pause");
    system("cls");
    goto bookmanagement;//返回第二级目录
    break;
```

按照格式，将书籍信息分条读入，再保存至本地 txt 即可

```
void library::addbook()//管理员权限：：添加书籍信息
{
    system("cls");
    booksnum++;
    cout<<"格式：书名 ISBN/ISSN 作者 分类号"<<endl;
    cout<<"example：理想国 978-7-80179-793-3 （古希腊)柏拉图著;吴献书译 B502.232/23"<<endl;
    cout<<"请输入题名（书名）:"<<endl;
    cin>>booksarray[booksnum-1].topic;
    cout<<"请输入ISBN/ISSN:"<<endl;
    cin>>booksarray[booksnum-1].type;
    cout<<"请输入作者:"<<endl;
    cin>>booksarray[booksnum-1].writer;
    cout<<"请输入分类号："<<endl;
    cin>>booksarray[booksnum-1].category;
    booksarray[booksnum-1].state=1;
    booksarray[booksnum-1].ownerid=0;
    ofstream ofs;
    ofs.open(FILEname,ios::out|ios::app);
    ofs<<booksarray[booksnum-1].topic<<" ";
    ofs<<booksarray[booksnum-1].type<<" ";
    ofs<<booksarray[booksnum-1].writer<<" ";
    ofs<<booksarray[booksnum-1].category<<" ";
    ofs<<booksarray[booksnum-1].state<<" ";
    ofs<<booksarray[booksnum-1].ownerid<<endl;
    ofs.close();
    cout<<"添加成功"<<endl;
    cout<<"即将返回上级目录"<<endl;
    return;
}
```

2. 删除图书信息

在上级目录“修改图书信息”下选择该功能后，自动调用 deletebook 函数，实现后返回清屏并返回上级目录“修改图书信息”。

```
case 2:
    hnu.deletebook();
    system("pause");
    system("cls");
    goto bookmanagement;//返回第二级目录
    break;
```

首先要找到该图书，然后再进行删除的操作

```
void library::deletebook()//管理员权限：：删除书籍信息
{
borrowfunc:
    system("cls");
    cout<<"你要删除哪本图书？请搜索"<<endl;
    this->xinxitypemenu();
    cout<<"输入信息的种类"<<endl;
    int xinxitype;
    cin>>xinxitype;
```

支持使用多种方式（共 4 种信息种类）查询该图书，下面仅以“标题”信息类为例，其余几种类别实现方式大致相同。

在搜索成功之后询问是否确定删除，若否则退出操作，若确定删除则将该书的各类信息赋为空，表示该图书已经被删除。

```
if(xinxitype==1){
    cout<<"请输入题名"<<endl;
    string topic_;
    cin>>topic_;
    int i=0;
    for(;i<this->booksnum;i++){
    if(topic_==this->booksarray[i].topic){
     cout<<"搜索成功"<<endl;
     this->showbook(this->booksarray[i]);
      if(this->booksarray[i].state==1){
          cout<<"你确定要删除该书吗？ 确认请输入1，取消请输入0"<<endl;
          bool borrow;cin>>borrow;
          if(borrow){
          cout<<"你已成功删除"<<" 《"<<this->booksarray[i].topic<<"》 "<<endl;
          this->booksarray[i].topic="";
          this->booksarray[i].type="";
          this->booksarray[i].writer="";
          this->booksarray[i].category="";
          this->booksarray[i].state=2;
          this->booksarray[i].ownerid=0;
          this->booksave();
          return;
          }else{
          cout<<"你已取消本次操作"<<endl;
          return;
          }
      }
    }}
    cout<<"找不到该书"<<endl;
    return;
    }
```

3. 修改图书信息

在上级目录“修改图书信息”下选择该功能后，自动调用 modifybook 函数，实现后返回清屏并返回上级目录“修改图书信息”。

```
case 3:
    hnu.modifybook();
    system("pause");
    system("cls");
    goto bookmanagement;//返回第二级目录
    break;
```

首先要找到该图书，然后再进行修改的操作。

```
void library::modifybook()//管理员权限：：修改书籍信息
{
borrowfunc:
    system("cls");
    cout<<"你要修改哪本图书？请搜索"<<endl;
    this->xinxitypemenu();
    cout<<"输入信息的种类"<<endl;
    int xinxitype;
    cin>>xinxitype;
```

支持使用多种方式（共 4 种信息种类）查询该图书，下面仅以“标题”信息类为例，其余几种类别实现方式大致相同。

在搜索成功之后询问是否确定修改，若否则退出操作，若确定则还需读入需要修改的信息的种类。

以下这段代码实现了上述的过程

```
if(xinxitype==1){
    cout<<"请输入题名"<<endl;
    string topic_;
    cin>>topic_;
    int i=0;
    for(;i<this->booksnum;i++){
    if(topic_==this->booksarray[i].topic){
     cout<<"搜索成功"<<endl;
     this->showbook(this->booksarray[i]);
        cout<<"你确定要修改该书吗？ 确认请输入1，取消请输入0"<<endl;
        bool borrow;cin>>borrow;
        if(borrow){
modifyfunc1:
            cout<<"信息格式："<<endl;
            cout<<"example：理想国 978-7-80179-793-3 (古希腊)柏拉图著;吴献书译 B502.232/23"<<endl;
            cout<<"请输入你要修改的信息种类"<<endl;
            xinxitypemenu();
            cout<<"*****  0.返回上级目录  *****"<<endl;
            cout<<"*****************************"<<endl;
            int option;
            cin>>option;
            system("cls");
```

在确定了修改信息的种类之后需要读入修改信息的结果并完成修改，即现在现行数组进行修改，再保存至本地 txt。

以下这段代码实现了上述的过程

```
308             switch(option)
309             {
310                 case 1:
311                         cout<<"请输入题名（书名）:"<<endl;
312                         cin>>booksarray[i].topic;
313                         this->booksave();
314                         cout<<"修改成功"<<endl;
315                         system("pause");
316                         goto modifyfunc1;
317                         break;
318                 case 2:
319                         cout<<"请输入ISBN/ISSN:"<<endl;
320                         cin>>booksarray[i].type;
321                         this->booksave();
322                         cout<<"修改成功"<<endl;
323                         system("pause");
324                         goto modifyfunc1;
325                         break;
326                 case 3:
327                         cout<<"请输入作者:"<<endl;
328                         cin>>booksarray[i].writer;
329                         this->booksave();
330                         cout<<"修改成功"<<endl;
331                         system("pause");
332                         goto modifyfunc1;
333                         break;
334                 case 4:
335                         cout<<"请输入分类号： "<<endl;
336                         cin>>booksarray[i].category;
337                         this->booksave();
338                         cout<<"修改成功"<<endl;
339                         system("pause");
340                         goto modifyfunc1;
341                         break;
```

```
342                 case 0:
343                         cout<<"即将返回上级目录"<<endl;
344                         system("pause");
345                         return;
346                 default:
347                         cout<<"输入格式错误，请重新输入"<<endl;
348                         system("pause");
349                         system("cls");
350                         goto modifyfunc1;
351                         break;
352             }
353         cout<<"你已成功删除"<<"《"<<this->booksarray[i].topic<<"》"<<endl;
354         this->booksave();
355         return;
356         }else{
357         cout<<"你已取消本次操作"<<endl;
358         return;
359         }
360       }
361     }
362     cout<<"找不到该书"<<endl;
363     return;
364     }
```

0. 返回上一级目录

在“修改图书信息”下选择该功能后，自动返回上级目录“管理员模式主菜单界面”。

```
case 0:
    cout<<"即将返回上级目录"<<endl;
    system("pause");
    system("cls");
    goto administrator_menu_func;//返回第一级目录
    break;
```

3. 搜索图书信息

以下代码实现了调用 search 函数来搜索特定图书的功能，功能完成后将返回上级目录。

```
case 3:
    hnu.search();
    system("pause");
    system("cls");
    goto administrator_menu_func;//返回第一级目录
    break;
```

由于在 search 的功能上，管理员所需要的功能与用户的功能大致相同，所以该函数将直接跳转至用户模块的搜索函数，具体代码以及过程详解详见用户模块。

4. 查看借阅情况

以下代码实现了调用 allborrowrecord 函数来展现所有已经被借阅的图书的功能，功能完成后将返回上级目录。

```
case 4:
    hnu.allborrowrecord();
    break;
```

以下代码实现了输出借阅日志（也就是所有借阅的记录）的功能

```
void library::allborrowrecord()
{
    for(int i=0;i<recordnum;i++){
       cout<<recordarray[i].ownerid<<" ";
       cout<<recordarray[i].topic<<" ";
       cout<<recordarray[i].writer<<" ";
       for(int j=0;j<booksnum;j++){
        if(recordarray[i].topic==booksarray[j].topic){
        recordarray[i].state=booksarray[j].state;}}
          if(recordarray[i].state==1){
          cout<<"已归还"<<endl;
         }else{cout<<"未归还"<<endl;}
    }
}
```

5. 查看馆藏图书列表

该功能可以查看本馆的所有藏书，进入该功能后，将调用 showbooklist 函数来展示所有书籍的信息，展示完毕后将返回上一级目录。

```
case 5:
    hnu.showbooklist();
    system("pause");
    system("cls");
    goto administrator_menu_func;//返回第一级目录
    break;
```

以下代码实现了管理员状态下所有书籍的信息的展示。

```
void library::showbooklist()//管理员权限：：展示所有书籍信息
{
    for (int i=0;i<=booksnum;i++)
    {
        book Book;
        Book=booksarray[i];
        if (Book.topic!="")
        {
            cout<<"该书的题名："<<Book.topic<<endl;
            cout<<"  该书的ISBN/ISNN："<<Book.type<<endl;
            cout<<"  该书的作者："<<Book.writer<<endl;
            cout<<"  该书的分类号："<<Book.category<<endl;
            if(Book.state==1) cout<<"  该书的借阅状况：未借出"<<endl;
            else if(Book.state==0){
            cout<<"  该书的借阅状况：已借出"<<endl;
            cout<<"  该书的借阅者："<<Book.ownerid<<endl;
        }
    }
    }
    return;
}
```

6. 查看用户列表

该功能可以查看本馆的所有用户，进入该功能后，将调用 showuserlist 函数来展示所有用户的信息，展示完毕后将返回上一级目录。

```
case 6:
    hnu.showuserlist();
    system("pause");
    system("cls");
    goto administrator_menu_func;//返回第一级目录
    break;
```

以下代码实现了管理员状态下所有用户的信息的展示。

```
26  void library::showuserlist()//管理员权限：：展示所有用户信息
27  {
28      for (int i=0;i<=usernum;i++)
29      {
30          if (userarray[i].account!=0) cout<<userarray[i].account<<endl;
31      }
32      return;
33  }
```

0. 退出登录

该功能可以注销管理员的登录状态，返回统一登陆界面。

```
case 0:
    cout<<"是否确认退出？"<<endl;
    cout<<"0：取消   1：确认"<<endl;
    int temp3;
    cin>>temp3;
    if (temp3==1)
    {
        cout<<"退出登录成功，即将返回登录界面"<<endl;
        system("pause");
        system("cls");
        goto sign;//返回登录界面
    }
    else
    {
        cout<<"已取消，即将返回上级目录"<<endl;
        system("pause");
        system("cls");
        goto accountmenufunc;//返回第二级目录
    }
    break;
```

四、小彩蛋-开发者信息展示

以下代码将实现前往开发者信息界面

```
278  case 0:
279      system("cls");
280      hnu.developerinfor();
281      system("pause");
282      cout<<"即将返回上级目录"<<endl;
283      system("pause");
284      system("cls");
285      goto sign;//返回登陆界面
```

以下为开发者信息的主界面

```
88  void library::developerinfor()
89  {
90      cout<<"*************************************************"<<endl;
91      cout<<"*****      恭喜成为第"<<1<<"位发现本系统的大佬      *****"<<endl;
92      cout<<"*****        《湖南大学图书馆管理系统》        *****"<<endl;
93      cout<<"*****       由湖南大学信息科学与工程学院       *****"<<endl;
94      cout<<"*****      计科2102班 黄政、梅炳寅 编写       *****"<<endl;
95
96
97      cout<<"*****鸣谢 星空不费电 粗心大法斩IDE的波波老师*****"<<endl;
98      cout<<"*****        和周二晚热心答疑谭彦恺助教       *****"<<endl;
99      cout<<"*****                                         *****"<<endl;
100     cout<<"*****            感谢您的使用与支持           *****"<<endl;
101     cout<<"*************************************************"<<endl;
102 }
```

五、关于谬误输入性息的处理

1、switch 分支的谬误输入性息处理

因为涉及到按键选择功能，所以本系统不可避免地涉及到许多分支，在这些分支的统筹上，采取 switch 语句来操作，但不可避免地有可能输入在允许范围之外的其他数字，此时我们设置了 default 语句并让程序返回该层初始状态，提示输入格式错误，并让操作者重新输入。

例如 user 目录下的 default 语句

```
101      default:
102          goto userfunc;
103          break;
```

又如管理员模式-账号设置下的 default 语句

```
default:
    cout<<"输入格式错误，请重新输入"<<endl;
    system("pause");
    system("cls");
    goto accountmenufunc;//返回第二级目录
    break;
```

再如管理员模式-书籍信息设置下的 default 语句

```
default:
    cout<<"输入格式错误，请重新输入"<<endl;
    system("pause");
    system("cls");
    goto bookmanagement;//返回第二级目录
    break;
```

再如管理员模式的 default 语句

```
default:
        cout<<"输入格式错误，请重新输入"<<endl;
        system("pause");
        system("cls");
        goto administrator_menu_func;//返回第一级目录
```

再如初始登录界面下的 default 语句

```
286 default:
287     cout<<"输入格式错误，请重新输入"<<endl;
288     system("pause");
289     system("cls");
290     goto sign;//返回登陆界面
```

这些 default 语句保证了程序的可延续性，避免了因操作不当产生的 bug 使程序轻易地崩溃，从而给用户带来不好的体验。

2、关于不存在的账户、书籍信息的处理

输入的该类信息首先将被纳入搜索中，如果在记录中没有成功搜索到，将返回相关的信息，避免用户不知情而继续产生错误的操作。

如在借书过程中未能成功搜索到该书

```
298     cout<<"找不到该书"<<endl;
299     return;
300     }
```

又如在管理员模式下搜索账户时未能成功找到目标账户

```
112         }
113         cout<<"未找到该账号"<<endl;
114         return;
115     }
```

再如在用户与管理员登陆时输入的账户或者密码出现问题

```
106 else if (hnu.login_user(a.account,a.password)==2){
107         system("cls");
108         cout<<"账号不符合格式，请重新登录"<<endl;
109         goto sign;//返回登陆界面
110         }
111 else if (hnu.login_user(a.account,a.password)==3){
112         system("cls");
113         cout<<"账号或密码错误，请重新登录"<<endl;
114         goto sign;//返回登陆界面
115         }
116         break;
```

这种情况下，我们设置了如下的两个函数来进行判断，分别对应不同的错误类型返回不同的值，再将操作者的错误种类输出，提示操作者改正。

```
 5  int library::login_user(int account,string password)//用户登录处理
 6  {
 7      ifstream ifs;
 8      ifs.open(FILENAME, ios::in);
 9      long account1;
10      string password1;
11      while (ifs >> account1 && ifs >> password1) {
12          if (account1==account)
13          {
14              if (password1==password) {return 1;/*账号、密码正确*/ }
15              else {return 3;/*账号正确，但密码错误*/ }
16          }
17      }
18      return 2; /*账号格式错误*/
19      ifs.close();
20  }
21
22  int library::login_admin(int account,string password)//管理员登录处理
23  {
24      ifstream ifs;
25      ifs.open(a_acount, ios::in);
26      long account1;
27      string password1;
28      while (ifs >> account1 && ifs >> password1) {
29          if (account1==account)
30          {
31              if (password1==password) {return 1;/*账号、密码正确*/ }
32              else {return 3;/*账号正确，但密码错误*/ }
33          }
34      }
35      return 2; /*账号格式错误*/
36      ifs.close();
37  }
```

3、关于重置密码时的验证

设置密码时，习惯上需要输入密码与确认密码，以保证密码的绝对准确性。但由于给出了初始密码，所以该步仅需要在重置密码时完成。

如管理员修改用户密码时的确认输入密码是否准确

```
82  void library::modifypassword()//管理员权限：：修改用户密码
83  {
84      long temp_account;
85      string temp_password,temp2_password;
86      system("cls");
87      cout<<"请输入要重置的账号:"<<endl;
88      cin>>temp_account;
89  repasswordfunc:
90      cout<<"请输入要重置的密码:"<<endl;
91      cin>> temp_password;
92      cout<<"请确认要重置的密码:"<<endl;
93      cin>> temp2_password;
94      if (temp_password!=temp2_password)
95      {
96          cout<<"密码不一致，请重新输入密码"<<endl;
97          system("pause");
98          system("cls");
99          goto repasswordfunc;
100     }
```

又如用户在重置密码时需要确认密码输入是否正确

```
40          system("cls");
41          cout<<"请输入新密码:"<<endl;
42          cin>>tempstr1;
43          cout<<"请确认你的新密码:"<<endl;
44          cin>>tempstr2;
45          if (tempstr1==tempstr2)
46          {
47           hnu.userarray[temp_account].password=tempstr1;
48          }
49          else goto again;
```

六、关于颜色的处理

在原程序的主函数上加上如下代码可以实现变色的效果。

```
 9  #include <windows.h>
10  using namespace std;
11  void colour(int x)
12  {
13      HANDLE h=GetStdHandle(STD_OUTPUT_HANDLE);
14      SetConsoleTextAttribute(h,x);
15  }
16  library hnu;
17  int main(){
18      colour(11);
```

4、分析与总结

1. 实验分析

1.本次综合实验我觉得理清思路，框架后其实并不难，与一些考数学的编程题不同，本次图书管理系统更注重将平时所学到的编程知识应用到实践中。

2.本次实验我们选择的数据结构是数组，复杂性不及链表等其他数据结构。但是对于本次所实现的相对较简单的功能来说，我认为使用数组已经游刃有余，十分简便，同时也便于维护、易于理解。

3.读写文件在本次实验中显得十分重要，为了使程序每次运行都能保留数据，需要用到<fstream>的读写文件知识。做到这点使我们初步体验到了程序员的成就感，更是这种沟通“黑屏”内外的力量给我们带来惊喜。能够在 txt 中做出改变，永久保存数据，而不仅仅是每次打开 dev 都要重置，这是一个巨大的飞跃，同时也是完成大型工程的必经之路，必备经验。

4.封装的思想是十分重要的，在本次工程编写中，会遇到许多可能有交集的、思想重复的代码，如果没有之前编写的代码的封装化与直接调用，代码量将大大加大，这对程序的实现与维护是大大不利的。同时结构体与类的大量运用，使数据与过程紧密结合，同时规范化，形式化，将相关的一类结合在一起，便于维护与代码的修改。

2. 遇到的问题

1. 构造函数读文件时，无文件只能创建一个文件。(想要两个文件)
2. 经常出现某个函数找不到定义，即无法关联的情况。
3. 以头文件或 cpp 的形式在工程文件夹下创建的内容无法与工程文件本身产生关联，会报“无法找到该变量/函数”的错误。
4. 修改完代码之后立即编译会很快通过，但并没有做出修改。

5. 在新建.h 或.cpp 之后会出现没有办法调用的现象。

3. 解决的办法

1. 第一个读文件时打开文件没有关闭文件。

2. 这是 dev 的 bug，每次修改代码后如果只编译而没有清楚则会导致出错，因此每次修改后需要清除一下或者关闭重开(因为这个问题浪费了很多时间)学习到的新知识点。

3. 需要在 dev 的 project 左栏中右击来创建，这样才可以正确关联与正确调用。

4. 这是 dev 在偷懒，此时需要点击 clean 来清楚已编译的数据，让 dev 重新编译，再次编译成功后即可操作。（摸索这一点也花了不少时间，甚至一直怀疑这是“魔法学院”的“魔法”）

5. 这是因为 main.cpp 内没有#include 其他的.h 文件。虽然很基础但是也着实坑了我们一把，这找不到错误的感觉令人心醉。

4. 学习到的新知识点

1.fstream 读写文件

2.提取一个字符串里某个区域之间的字符串

(三级分类搜索图书时要用到)

3.system 相关操作指令 system(“pause”)和 system(“cls”)

这些操作是必要的，在显示方面对用户比较友好。

4.“goto sign1”与“sign1：”，这种代码中跳跃的方式虽然不推荐在平时的课内学习中使用，但真的用在实践中是十分方便的。不然没法解决返回上一级目录的问题。

5.HANDLE h=GetStdHandle(STD_OUTPUT_HANDLE);实现变色的效果。