

大类基础课程大型实验

**2014/2015(2)**



实验题目 图书管理系统

学生姓名 王程飞

学生学号 20186061219

学生班级 软件工程05

任课教师 毛国红

提交日期 2019.5.21

**计算机科学与技术学院**

**图书管理系统 实验报告**

1. **大型实验的内容**

合同管理系统（CMS：Contract Management System）用于合同管理，包括生成合同，修改合同，删除合同，按要求查询合同，文件读取等功能。要求使用学习过的C/C++程序设计的知识完成合同管理系统的设计与实现。

1. **运行环境**

合同管理系统（CMS）在Dev C++ 平台下开发，操作系统：Windows 10。

硬件环境：

处理器：Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz

内存：8.00GB

系统类型：64位操作系统

1. **实验课题分析（主要的模块功能、流程图）**

**3.1 图书管理系统的主要功能**

**合同管理系统（CMS）主要功能为：**读者管理、工作人员管理、管理人员管理，可以完成借书、还书、查找图书信息、维护图书、人员信息等工作。详细的系统功能结构为图1所示。

合同管理系统（LMS）

增删模块

查询模块

修改模块

文件读写模块

图1 系统结构图

**系统各模块的功能具体描述为：**

1. **文件读写模块**

打开文件读取合同信息，并将改动保存到文件

1. **增删模块**

新增合同和删除合同

1. **查询模块**

按日期查询合同，显示所有合同，查询某一天的合同，到期合同功能

1. **修改模块**

修改指定合同的ID, 甲方, 乙方, 时间, 金额, 内容等.

**3.2 系统分析及设计**

**系统涉及对象有两个基本类：**时间类和合同类。其中合同类涉及的功能操作归纳为如下表1所示：

表1 合同类的操作

|  |  |
| --- | --- |
| **对象** | **涉及的对象操作** |
| 时间 | 输出时间 |
| 输入时间 |
| 验证输入合法性 |
| 比较大小 |
| 合同类 | 合同开始时间 合同结束时间 合同签订 |
| 合同ID |
| 甲方 乙方 |
| 合同金额 |
| 合同内容 |
| 输入合同，输出合同 |

采用面向对象的方式实现合同管理系统。系统的主要的类结构如图2所示。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 类/成员 | 1 | 2 | 3 | 4 | 5 |
| Date |  |  |  |  |  |
| Contract |  |  |  |  |  |
| ContractUI |  |  |  |  |  |

图2 系统主要类结构图

分别设计读者类、工作人员类、管理员类，人员类为基类实现通用的功能。

用文本文件进行数据的保存，需要保存的数据主要包括图书数据、用户数据（包括读者、工作人员、管理员）、借阅索引表。设置数据操作类，实现所有的文本操作相关的功能。

**3.3系统的实现**

**（1）类的编写**

系统工程名为：LMS。包含了People类（所有人员的基类），Book类（图书类）两个基本类，由于系统中需要三种使用对象，所以从People类派生出三种类型的用户，即Reader类（读者类），Worker类（工作人员类）及Manager类（管理人员类）。而这些人用户都有相应的用户名name和编号no，运用时将三个派生类中的用户名和用户密码分别与name和no对应。

具体类结构声明如下：

* **People类：**

class People{//基类

public:

string name;//姓名

string no;//账号

People(string na="null",string n="null"):name(na),no(n){}

string mpeople(string a)//改名字

{

return name=a;

};

void display(){cout<<"姓名："<<name<<" "<<" 账号："<<no<<endl;}

};

* **Book类**

class Book{

private:

string bno;//书号

string bname;//书名

string author;//作者

string press;//出版社

int storenum;//藏书量

int borrownum;//借出数

int canborrow;//可借数量

public:

Book(string bno1="null",string bname1="null",string author1="null",string press1="null",int storenum1=0,int canborrow1=0,int borrownum1=0)

:bno(bno1),bname(bname1),author(author1),press(press1),storenum(storenum1),canborrow(canborrow1),borrownum(borrownum1){}

Book \*next;

Book\* getnext()const{return next;}

string gbno()const{return bno;}//获得书号

string gbname()const{return bname;}//获得书名

string gauthor()const{return author;}//获得作者

string gpress()const{return press;}//获得出版社

int gstorenum()const{return storenum;}//获得藏书量

int gcanborrow()const{return canborrow;}//获得可借数量

int gborrownum()const{return borrownum;}//获得借出数量

bool operator==(const Book& m){return(bno==m.bno);}

void bookdisplay();

void storebook(){storenum++;canborrow++;}//存入该书

void returnbook(){//还书

canborrow++;

borrownum--;

}

void borrowbook(){//借书

canborrow--;

borrownum++;

}

void setnull(){bno="null";}//将书号设置为null

bool ifbnofull();//判断书号是否为null

void setinformationg();//设置图书信息

void setbno(string str){bno=str;}

void setauthor(string str){author=str;}

void setpress(string str){press=str;}

};

**…….//罗列所有类的声明**

* **Reader类：**
* **Manager类：**
* **Worker类：**

**（2）链表的使用**

系统实现采用文件的输入输出流对文本数据进行读取与写入，但是由于读者信息、图书信息、工作人员信息以及管理员信息都是一个数据的集合，于是对数据的存储组织使用了单向链表。

因为图书管理系统在登录、查找、修改、添加的时候都需要处理大量的数据，所以使用链表十分必要。以图书信息为例，在Book类的基础上定义一个对应的Booklist类来管理图书数据，具体的结构声明如下：

class Booklist{

int size;

Book\* ptr;

public:

Booklist():ptr(0),size(0){}

void remove(Book& rea);//移除

void add(Book& a);//添加信息到链表中

void booklistdisplay();

void addbook(Book& a);//图书库增加新书

void find(int i);//查找

void write();//写入文件

void readerbookfind(string are);//读者信息中图书查找

Book\* allfind(Book& finbook);

void stringremove(string str);//依靠字符来移除信息

void advancedfind();

~Booklist();

};

在运用时，令当前图书的next结点指向新的图书结点，即结点的指针next保存新的图书结点的地址（如下图3所示），以此类推，所有图书信息就通过链表的形式串联起来了。

Book1 Book2 Bookn

data

data

data

......

\*next

\*next

\*next

图3 图书链表的建立

读者（Reader）、工作人员（Worker）及管理员（Manager）的链表设置与图书的类似，也是通过定义类的指针变量，通过指向下一个类的地址将信息串联起来。即在Reader的基础上定义Readerlist，在Worker的基础上定义Workderlist，在Manager的基础上定义Managerlist。各个类的结构声明如下：

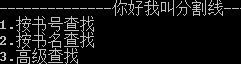
**…….//罗列所有类的声明**

* **Readerlist类：**
* **Managerlist类：**
* **Workerlist类：**

图书管理系统的信息的管理就具体表现为链表的操作。拿图书信息来说，图书信息的查找、修改、添加和删除与链表的查找、修改、添加、删除对应。

* **图书的查找：**

在用户在查询图书时，有三种不同的模式，见图4：

 图4 图书查找界面

这三种情况都需要对链表中的所有数据进行顺序的搜索。 例如书名的查找，定义一个Book类指针变量r并对其初始化，使其为ptr。



从第一个Booklist类的第一个节点开始，会将文档里的书号bname和用户输入的书名比较，不一致时，r=r->next，与第二个数据对比，以此类推，直到找到相同的书名，此时调用函数r->bookdisplay()输出关于该书籍书籍的所有信息，即它的7个变量的数据。

如果没有找到可以匹配的，则cout<<"没有找到相关图书信息"<<endl；并结束循环。

**图书查找的流程图如下（备注：选择合适的工具绘制程序流程图）：**

是

是

是

否

用户选择一种查找模式

显示查找菜单

用户输入相应查找线索

用户选择合理？

否

提示选择错误

结点信息未匹配

且链表未结束？

图书链表遍历指针初始化

输出找到的图书信息

否

遍历到下一个结点

结点信息匹配？

提示找不到该图书

**…….//采用相仿的模式描述其他链表操作的实现**

* **图书的修改：**
* **图书的添加：**
* **图书的删除：**

**（3）交互界面以及登录菜单的实现**

系统运行开始的界面如图5所示：

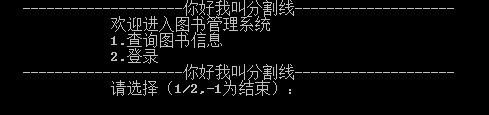


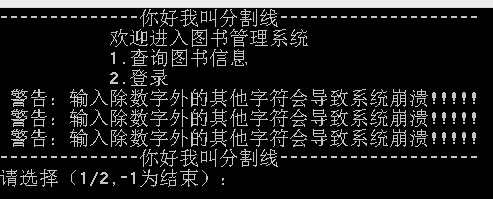
图5 开始登录界面

主要通过选择结构和循环结构实现界面的前进和后退。例如，第一个登录界面出现3个选择：1.查询图书信息，2.登录，-1.结束用switch case分别实现，选择之后转到下一个界面。

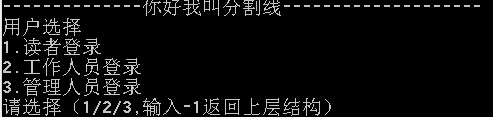
1. **实验调试、测试、运行记录及分析**

系统在调试测试过程中遇到若干问题，不过经过仔细反复的检查已经消除各种bug。

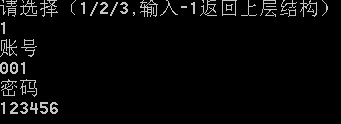
主要的测试经过如下：



在开始界面输入“2”即可跳转到登陆界面。



然后输入“1”，屏幕上跳出读者界面。

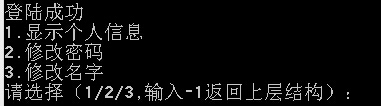


后输入读者信息，如：

读者编号：001

密码：123456

可跳出读者的个人界面



若执行“1”，则显示个人信息。



若执行“2”，则可修改密码。

C:\Users\wcf\Users\wojiaojzy\AppData\Roaming\Tencent\Users\1055422467\QQ\WinTemp\RichOle\66X]}F{A}UY[M[RFYN0%@TN.jpg

然后输入修改后的密码即可。

若执行“3”，则可修改名字。

C:\Users\wcf\Users\wojiaojzy\AppData\Roaming\Tencent\Users\1055422467\QQ\WinTemp\RichOle\W@)PSVFN)(AM@2X)3QCV2@R.jpg

然后输入修改后的名字即可。

**遇到的问题及解决方法如下：**

* + **问题1：**

**问题描述：**开始采用二进制进行文件数据的读取，想让用户输入的账户密码与内存从文件中读入的数据相匹配，但是一直是登录失败，即使输入的是正确的账号。

**解决方法：**我想将内存中获取的数据显示出来以便比对，于是设计了划红线的函数（见图6），想让它输出预设的账号，但发现它却把所有的数据（账号 姓名 密码）都输出了。于是检查到原来程序从文件中读入的数据并没有与原来构造函数工作一致：账号关联num，姓名关联name。类似的错误也在写文件中发生，写入的文本中还会出现：烫烫烫烫烫。重新修改读写代码后就正确了。

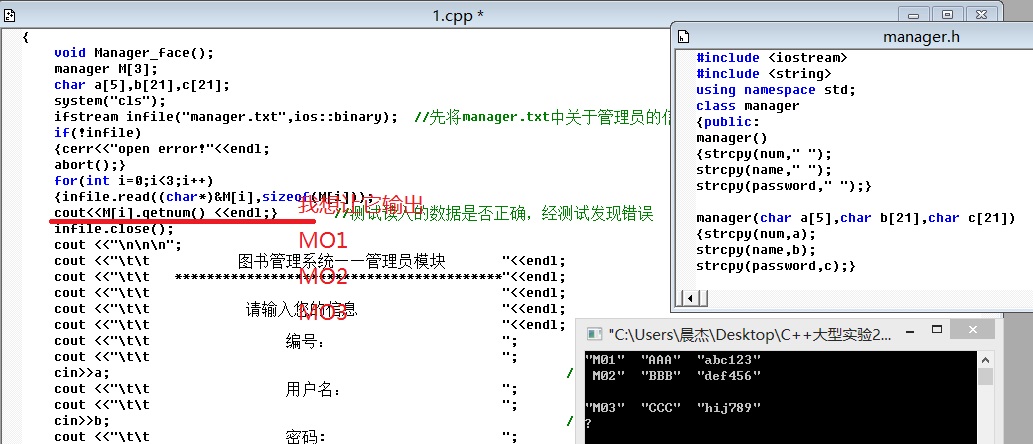


图6 调试测试问题1

**…….//采用相仿的模式描述其他调试测试问题**

* + **问题2：**
  + **问题3：**
  + **问题4：**
  + **问题5：**

**……**

1. **实验总结（优点、不足、收获及体会）**

我设计的图书管理系统基本满足任务书的功能要求，类的结构和关系清晰，功能完善，而且在批量数据存储上采用了链表，在空间的利用率、集合数据操作（添加和删除）上具有优势。也能支持多种模式的数据查找。

存在的缺点主要是对于数据管理方面，针对4类对象：图书、读者、工作人员、管理员我设计了4个对应的链表类，这个设计后来仔细考虑实际是有冗余的，可以通过设计一个链表的模板类来解决冗余。另外，链表在查找上好像没有什么优势，无论用户给出什么样的查找条件我都采用了从头开始遍历链表的方法进行查找，不能达到有序数组里的二分搜索这样的高效率，希望在今后的学习中能找出基于链表的高效查找方法。

通过这次C++的大型实验，我深刻的明白到：课本知识与实践能力相结合的重要性。会读程序的人并不一定会编程序。要想把一门专业课程学好，必须增强自己的动手实践能力，而不是一天到晚只知道看书，那种行为只不过是“纸上谈兵”。看再多的书都不如自己亲手试一试。俗话说的好：会打仗的士兵才是好士兵。再者，课本上的知识不一定是完全准确的，只有自己动手进行试验过才知道。实践是检验真理的唯一标准，这话不假。

1. **附录：源代码**