Project 1 技术报告

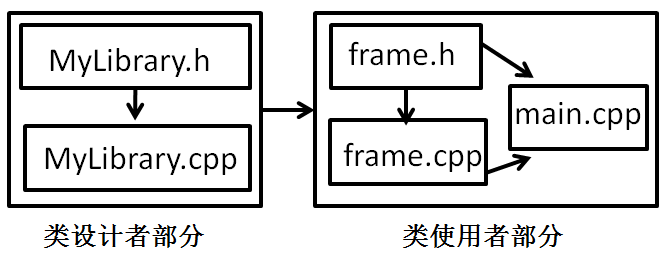
姓名：颜彬 学号：16337269班级：教务4班

* 需求分析：

图书管理系统功能如下：

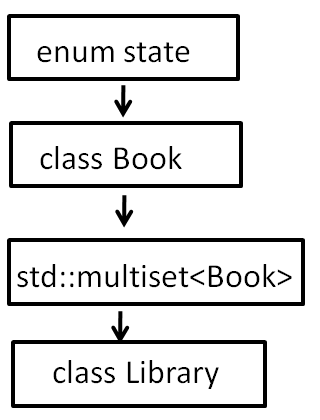
1. 添加书籍。
2. 删除书籍。当系统中存在此书时，删除该书籍，否则提示用户删除失败。
3. 借阅书籍。当系统中存在此书且未被借阅时，将书的状态设置为“借出”，否则提示用户借阅失败。
4. 归还书籍。当系统中存在未归换的该书时，将书的状态设置为“可用”，否则提示用户归还失败。
5. 查询全部。输出系统中所有的书籍（书名升序）以及他们的状态。
6. 查询书名。输出系统中所有与书名同名的书以及他们的状态。
7. 书籍计数。输出系统的藏书量或某一本书的副本数量。

* 文件间关系



1. MyLibrary.h与MyLibrary.cpp。定义并实现了Book类和Library类。前者用以表示书籍，后者构成整个系统。这部分属于类的设计范畴。（注：**类设计者部分没有实现输出，输出由“使用者”实现**）。
2. frame.h与frame.cpp。定义并实现了在main.cpp中使用了的函数。这些函数全部与输出有关。
3. main.cpp。程序运行的入口。直接调用library库和frame的函数。此文件内实现了简单的交互界面。
4. 主要工作委托给std：：multiset实现。

* 数据设计



1. State 枚举
   1. enum State {Borrowed, Available}; //Borrowed表示被借出，Available表示书籍可用。
2. Book 类

class Book {

public:

//定义类型别名以实现抽象

typedef multiset<Book> data\_t;

//不允许书籍默认构造。只允许通过传入书名构造。将书籍状态设为“可用”

Book() = delete;

Book(const name\_t& \_name) : name(\_name), state(Available) { }

~Book() = default;

//使用书名比较书籍间的相互关系

bool operator<(const Book& other) const { return name < other.name; }

bool operator==(const Book& other) const { return name == other.name; }

//一些访问和修改的函数

//关于书籍名字和书籍状态（“可用”，“已借”）

State GetState() const { return state; }

name\_t GetName() const { return name; }

void SetName(const name\_t& \_name) { name = \_name; }

void SetState(State newState) { state = newState; }

private:

name\_t name;

State state;

};

1. std::multiset<Book>用以储存数据
2. Library类

class Library {

public:

//定义一系列类型类型别名实现抽象

typedef multiset<Book> data\_t;

typedef typename Book::name\_t name\_t;

typedef typename data\_t::iterator Iterator;

typedef vector<Book> Info\_t;

typedef string file\_name\_t;

//只允许该类默认构造，不允许复制

Library() = default;

Library(const Library&) = delete;

Library& operator=(const Library&) = delete;

//分别为增加，删除，借阅，归还书籍。

void Insert(const Book& book) { libraryData.insert(book); }

bool Remove(const name\_t& name);

bool Borrow(const name\_t& name);

bool GiveBack(const name\_t& name);

//返回该类中书籍的相关信息

vector<Book> GetBookInfo(const name\_t& name) const ;

vector<Book> GetAllBookInfo() const;

//返回该类中书籍的数目

size\_t Count(const name\_t& name) const { return libraryData.count(Book(name)); }

size\_t NumOfBook() const { return libraryData.size(); }

//保存信息到文件，从文件读取信息

//接受文件名为参数

void SaveToFile(const string& fileName) const;

bool LoadFromFile(const string& fileName);

private:

//储存书籍的容器

data\_t libraryData;

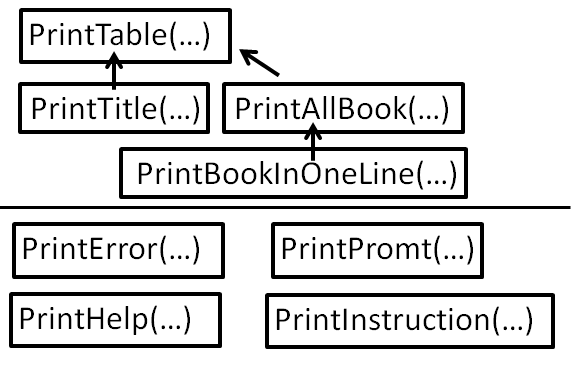
//内部辅助函数。输入书籍的名字，返回一对迭代器（std::pair）指向一个区间范围。该范围内的书籍都有相同名字name

inline auto \_find(const name\_t& name) const -> const pair<Iterator, Iterator>

{ return libraryData.equal\_range(Book(name)); }

};

* 函数间关系（指frame文档内的函数）



说明：

1. 所有函数的第一个参数都为ostream& os，用以接受输出流。在main中都将该参数赋为cout。调用者可根据实际情况修改。
2. 所有函数都返回void。
3. 下层的四个函数相互独立。
4. inline void PrintPromt(ostream& os);

//打印“>>> ”用以实现交互界面。

1. void PrintInstroduction(ostream& os);

//打印指导信息。每次运行该系统都会输出该内容。

1. void PrintHelp(ostream& os);

//打印帮助信息。在系统中输入help会调用该函数输出帮助。

1. inline void PrintError(ostream& os, const string& s)；

//打印错误信息。该函数会将参数s直接打印出来（加上换行符）。

1. inline void PrintTable(ostream& os, const Library& lib)

//将系统中的全部书籍以表格形式输出。接受library类。通过调用PrintTitle（打印表格）和PrintAllBook（打印所有书籍）执行打印。

1. inline void PrintTitle(ostream& os)

//打印表格的题头。执行实际的打印行为。

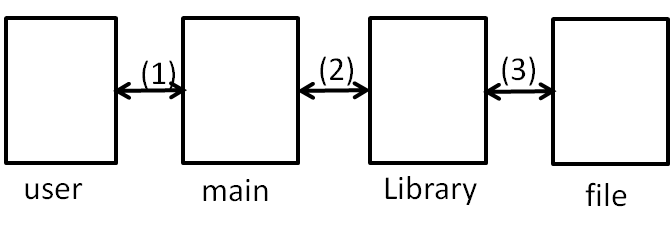
1. void PrintAllBook(ostream& os, const Library& lib);

//打印系统中的全部书籍。通过调用PrintBookInOneLine打印书籍。

1. inline void PrintBookInOneLine(ostream& os, const Book& book)

//实际执行打印行为的函数。每次只打印一行。接受Book的对象为参数。

* 数据交流



注：

user指这个程序实际的用户（甲方）。

main指Library class的使用者。

Library指Library class的底层实现。

file指储存信息的本地文件

**不提供跨层交流数据的接口。**如main向file交换数据必须通过library。

1. 从user到main。通过cin交换信息。
2. 从main到user。通过frame.h和frame.cpp中定义的一系列函数进行输出到终端。（见函数间关系）
3. 从Library到main。

vector<Book> Library::GetBookInfo(const name\_t& name) const ;

vector<Book> Library::GetAllBookInfo() const;

通过Library类中定义的两个函数传送书籍信息到main。

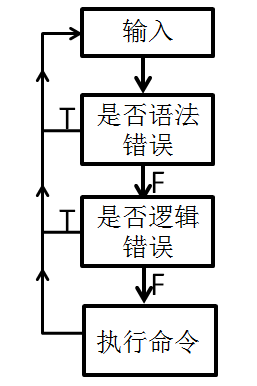
以及通过Library成员函数的返回值传递信息。

1. 从main到class。

通过Library类中的成员函数的参数传递来传递数据。

1. 从Library到file。main向Library提供文件名，library根据该文件名存放数据入file中。
2. file到library。main向library提供文件名，library根据该文件名向file读取数据。

* main函数的算法流程：



1. 输入：输入使用getline读取整行。
2. 是否语法错误：使用正则表达式检查语法错误。若格式不匹配，输出错误提示信息，重新等待输入。
3. 是否逻辑错误：调用Library类并查看返回值。若返回false表示出现逻辑错误。输出提示信息，重新等待输入。
4. 执行命令：命令被执行。

* 合法的命令如下

1. add <bookname>：添加一本书。
2. rm <bookname>：删除一本书
3. brw <bookname>:借出一本书
4. ret <bookname>:归还一本书
5. cnt <bookname>：统计该书的数目
6. cnta：统计所有（all）书的数目
7. prt <bookname>：打印该书相关信息
8. prta：打印所有书的信息
9. save：将信息储存到本地文件
10. load：载入本地文件中的信息

* 使用的正则如下：

1. 匹配命令的格式：^(quit|help|save|load|prta|cnta)(\\s.\*)?$|^(add|rm|brw|ret|cnt|prt)
2. 匹配整个输入的格式：

^(quit|help|save|load|prta|cnta|(add|rm|brw|ret|cnt|prt) [a-z\\sA-Z\_]+(\\w|\_)\*)$