Bookstore 开发文档

项目名称

Bookstore

文档作者

Limer

程序功能概述

维护一个书店管理系统

- 可进行命令行交互
- 维护一个账户系统
- 维护一个图书系统
- 维护一个日志系统
- 性能良好

主题逻辑说明

命令读入交由 *Command* 处理，*Command* 分发给各个 *Class* ， *Class* 从外存中读取数据，执行命令返回相关值，并写入数据。

代码文件结构

```
src
+-- main.cpp
+-- Command.h
+-- User.h
|   +-- User
|   +-- UserSystem
+-- Book.h
|   +-- Book
|   +-- BookSystem
+-- Log.h
| +-- Staff
| +-- Finance
|   +-- Log
+-- BlockList.h
|   +-- BlockNode
|   +-- BlockList
+-- File.h
```

接口说明

```
// Command.h

// implement command, deliver to different classes and functions
bool Input(string command);

// exit the system
bool Quit();

// quit the system
bool Exit();
```

```
// User.h -> class User

// ID: letter, number and _; max length = 30
char ID[];

// name: visible ascii; max length = 30
char name[];

// password: letter, number and _; max length = 30
char password[];

// priority: {7, 3, 1}
int priority;
```

```
// User.h -> class UserSystem

// Use the blocklist and file system to record the users

// the stack of users
stack<User> users;

// transfer to specified user
bool Su(string user_ID, string password = "");

// log out
bool Logout();

// register
void Register(string user_ID, string password, string user_name);

// change the password
void Passwd(string user_ID, string new_password, string old_password = "");

// add the user
```

```
void UserAdd(string user_ID, string password, string priority, string user_name);

// delete the user
void Delete(string user_ID);
```

```
// Book.h -> class Book

// ISBN: visible ascii; max length = 20
char ISBN[];

// name: visible ascii except ""; max length = 60
char name[];

// author: visible ascii except ""; max length = 60
char author[];

// n: the number of keyword
int n;

// keyword: visible ascii except ""; max length = 60; divided by |
char keyword[][][];

// price: number and .; max length = 13; 2-bit precision
double price;

// quantity: number; max length = 10; int
int quantity;

// total_cost: number and .; max length = 13; 2-bit precision
double total_cost;
```

```
// Book.h -> class BookSystem

// use the blocklist and the file system to record the books, multiple index files and
a single data file
// return true for successful operation, false for failed one

Book selected;

bool ShowISBN(string ISBN, vector<Book> &res);

bool ShowName(string name);

bool ShowAuthor(string author);

bool ShowKeyword(string keyword);

bool ShowAll();
bool buy(string ISBN, int Quantity);

bool Select(string ISBN);

bool Modify(string ISBN, string name, string author, string keyword, string price);

bool Import(int quantity, double total_cost);
```

```
// Log.h -> class Staff;

// the same format of the input command

bool Report(string name);
```

```
// Log.h -> class Finance;
 ----
|
bool ShowFinance(int time = -1);

bool Report();

/*
the format of financial statement
--------------------------------------------------
Book            Quantity  Total cost
--------------------------------------------------
Hamlet            100     +10000
All's Well That Ends Well   -200    -100000
--------------------------------------------------
*/
```

```
// Log.h -> class Log;
// the same format as input command
```

```
// BlockList.h -> Class BlockList

// define a MINBLOCKSIZE and a MAXBLOCKSIZE
// add the second word to make sure every element is unique to reduce the time
complexity and corner cases
// when a block size is greater than MAX, split it
// when a block size is smaller than MIN, merge it with adjacent block
// insert when there is no element in any block, create one and insert element into it
// delete when there is only one element in the last block, remove the block, thus no
block remained
// find the smallest index of the key, scan from it to find the set of element with
same key
```

```
int NextBlock(const int &offset);

void DeleteBlock(const int &offset);

void MergeBlock(const int &offset1, const int &offset2);

void SplitBlock(const int &offset);

void FindNode(const std::string &key, std::vector<int> &array);

void AddNode(const Node &node);

int DeleteNode(const Node &node);
```

```
// File.h -> class File

// simple read and write

template<class T, int n = 2>

void ReadInfo(int value, int index);

void WriteInfo(int value, int index);

void Read(T &value, int index);

void Write(T &value, int index);
```

存储说明

仅 *book selected, stack users* 在内存存储，其他尽量在外存存储。

具体算法说明

块状链表

- 仅维护索引，用 *File.h* 来额外维护数据记录
- 对于重复键值，可记录第二关键字使其键值不重复
- 注意插入第一个元素，插入最后一个元素等 *corner cases*

其他补充说明

- 模板类的成员函数尽量少
- 文件读写尽量定长，尽管不定长亦可，当然这个并不需要多在意
- 尽量减少文件读写，提高复用率