

河南工业大学

课 程 报 告

课程名称: 面向对象程序设计

专业班级: 软件 1305 班

小组成员: 田劲锋 201316920311

邢志鹏 201316920311

王增辉 201316920311

任课教师:

完成日期: 2015 年 1 月 13 日

学 期: 2014-2015 学年第一学期

目录

1	题目内容及设计要求	2
2	总体方案设计	3
2.1	总体功能框图	3
2.2	类的设计说明	3
2.2.1	Number 类	3
2.2.2	Score 类	5
2.2.3	UI 类	7
2.3	主要算法流程图	9
3	程序清单及注释	11
4	运行结果与分析	12
5	总结	13
A	股票交易系统	14
A.1	设计类图	14
A.2	题目内容	15
A.3	头文件	15
	参考文献	21

1 题目内容及设计要求

猜数字游戏

内容及要求：

猜数：用户从键盘输入4位不重复的数，来匹配计算机给出的4位随机数，若数字和位置均等同，表示用户赢了。每猜一次，计算机均给出提示信息 (x, y) ， x 表示数字、位置都匹配的个数， y 表示数字匹配但位置不匹配的个数。

- (1) 设计有好的中文交互界面；
- (2) 按8888键，可以得到更详细的帮助信息，如：第1位数字正确等。
- (3) 按7777键后，可以查看计算机所给的4位数，但需要输入密码，密码自定。
- (4) 猜的结果以分数给出，每猜错一次扣40分，若猜对1个数，奖励20分。
- (5) 每次游戏结束后将分值存盘，文件名自定。

难度系数：

1.1

以上题目要求解决问题步骤为：

- (1) 应用系统分析，建立该系统的功能模块框图；
- (2) 分析系统中的各个实体及它们之间的关系；
- (3) 根据问题描述，设计系统的类层次；
- (4) 完成类层次中各个类的描述；
- (5) 完成类中各个成员函数的定义；
- (6) 完成系统的应用模块；
- (7) 功能调试；
- (8) 完成系统总结报告。

2 总体方案设计

2.1 总体功能框图

考虑猜数功能的实现，为了实现最大程度的解耦，将与用户交互的操作全部封装在 UI 类中。

考虑对于全局唯一的用户，需要一个全局唯一的计分器，这里设计为单例模式的 Score 类。

考虑到每次运行程序用户会进行多次猜数，对于每次猜数都实例化一个 Number 对象，并提供对猜数的检查功能。

图 1 展示了各个类之间的关系。

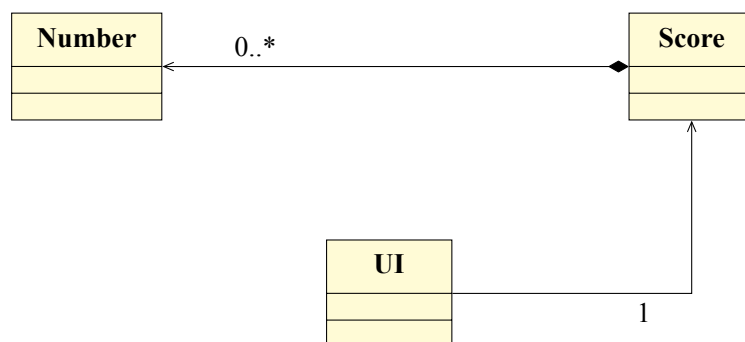


图 1: 猜数字游戏各个类之间的关系

2.2 类的设计说明

2.2.1 Number 类

Number	
# number: int	//待猜的数
- numbers: short[]	//数的拆分数组
# count: int	//猜数次数
+ Number()	//初始化产生随机数
- genRand(): int	//产生符合要求的随机数
+ guess(number: int): x: int, y: int	//猜数并返回提示信息
+ detail(number: int): int[]	//猜数并返回正确的位数组
+ answer(): int	//返回正确的答案
+ ~Number()	//析构函数

图 2: Number 类

Number::Number

Number::Number();

返回 构造函数

描述 调用 `Number::genRand()` 来产生符合要求的随机数，调用 `this→setNumber()` 来设置 `this→number` 和 `this→numbers`。初始化 `this→count` 为 0。

Number::genRand

int Number::genRand();

返回 生成的随机数

描述 生成符合要求的随机数。算法见第 9 页的图 5。

Number::setNumber

void Number::setNumber(int number);

返回 无

描述 将 `number` 放置到 `this→number` 里，并将 `this→numbers` 置为对应的拆分数组，这两个是一一对应的关系，不可分拆，只是单纯的方便后面的计算。

Number::guess

std::pair<int, int> Number::guess(int number);

返回 提示信息 (x, y)

描述 猜数。用户猜的是 `number`，判断是否正确并返回提示信息 (x, y) ， x 表示数字、位置都匹配的个数， y 表示数字匹配但位置不匹配的个数。全部正确时应该返回 $(4, 0)$ 。每调用该方法时 `this→count` 应该自增 1。算法见第 10 页的图 6。

Number::detail

std::vector<int> Number::detail(int number);

返回 详细信息的数组

描述 对应 8888 键的功能。用户上一次猜的是 `number`，返回一个数组，数组的每个元素表示哪一位正确了。全部正确时应该返回 $\{1, 2, 3, 4\}$ 。每调用该方法时 `this→count` 应该自增 1。算法见第 10 页的图 7。

Number::answer

int Number::answer();

返回 正确答案

描述 对应 7777 键的功能。返回正确答案的 `this→number`。每调用该方法时 `this→count` 应该自增 1。

2.2.2 Score 类

Score	
# PLUS: <i>int</i> = 20	//加分分值
# MINUS: <i>int</i> = 40	//减分分值
# score: <i>int</i>	//得分
+ <i>getScore()</i> : <i>int</i>	//获取得分
# <i>lastNumber</i> : <i>int</i>	//用户上一次猜的数
# <i>numbers</i> : <i>Number</i> []	//存储每次猜数对象
- <i>password</i> : <i>string</i>	//密码
- <i>Score()</i>	//构造函数
+ <i>getInstance()</i> : <i>Score&</i>	//获取实例
+ <i>~Score()</i>	//析构函数
+ <i>newGame()</i> : <i>void</i>	//新建游戏
+ <i>newGame(number: int): void</i>	//指定数字新建游戏
+ <i>guess(number: int): bool, string</i>	//猜数
# <i>plus()</i> : <i>void</i>	//加分
# <i>minus()</i> : <i>void</i>	//减分
+ <i>read()</i> : <i>int</i>	//读入分数
+ <i>write()</i> : <i>void</i>	//写出分数
+ <i>checkPassword(password: string): bool</i>	//检查密码

图 3: Score 类

Score::Score

Score::Score();

返回 构造函数

描述 单例模式的构造函数是 **private** 的。构造函数尝试调用 **this→read()** 从文件中读取分数和密码，初始化 **this→score** 和 **this→password**。

Score::getInstance

static Score& Score::getInstance();

返回 唯一的 Score 实例引用

描述 单例模式中，获取唯一的 Score 实例。这是个静态方法。

Score::~~Score

Score::~~Score();

返回 析构函数

描述 调用 **this→write()** 向文件中写出得分。释放所有申请的动态内存。

Score::newGame

void Score::newGame();

返回 无

描述 创建一个新的 Number 实例，添加到 **this→numbers** 数组的尾部。

Score::newGame

```
void Score::newGame(int number);
```

返回 无

描述 以指定的 `number` 为参数，创建一个新的 `Number` 实例，并添加到 `this→numbers` 数组的尾部。

Score::guess

```
std::pair<bool, std::string> Score::guess(int number);
```

返回 是否猜对以及提示信息

描述 用户输入了 `number`。

对于特殊情况，如果 `number` 是 8888，则调用当前 `Number` 对象（即 `this→numbers` 数组的最后一个元素）的 `detail()` 方法，参数为 `this→lastNumber`，并返回猜数失败和详细的帮助信息，即“第 *a, b, c* 位数字正确”；如果是 7777，则调用 `answer()` 方法来查看答案，并返回猜数失败和正确答案，即“正确答案是 *number*”。特殊情况不加减分数。

对于一般的猜数，则调用 `guess()` 进行正常的猜数流程，如果猜对则返回猜数成功和祝贺信息；猜错则返回猜数失败和提示信息 (x, y) ，即“数位匹配 x 个，数匹配位不符 y 个”。猜对的加分，猜错的则减分。

Score::plus

```
void Score::plus();
```

返回 无

描述 给 `this→score` 加上 `Score::PLUS` 的分值。

Score::minus

```
void Score::minus();
```

返回 无

描述 给 `this→score` 减去 `Score::MINUS` 的分值。

Score::read

```
int Score::read();
```

返回 读入的得分

描述 从文件中读取得分和密码，分别放到 `this→score` 和 `this→password` 里。

Score::write

void Score::write();

返回 无

描述 向文件中写出得分 this→score 和密码 this→password。

Score::checkPassword

bool Score::checkPassword(std::string password);

返回 密码是否正确

描述 对比 password 和 this→password 是否一致。

2.2.3 UI 类

«utility» UI	
+ Main(): void	//主循环
+ MainMenu(): int	//主菜单
+ NewGame(): void	//新游戏
+ GuessNumber(): bool	//猜数字
+ ViewDetail(): void	//8888
+ ViewAnswer(): void	//7777
+ ShowScore(): void	//显示得分
+ InputPassword(): bool	//输入密码

图 4: UI 类

UI::Main

void UI::Main();

返回 无

描述 循环调用 UI::MainMenu()，根据其返回值调用 UI::NewGame() 或者 UI::ShowScore()。直到其返回 0 表示退出，此时中止循环。

UI::MainMenu

void UI::MainMenu();

返回 无

描述 显示主菜单，等待用户输入选项，输入后返回选项值。

定义: (1) 新游戏; (2) 显示得分; (0) 退出。

对于用户的其他不合理输入，一律解析为 0，表示退出。

UI::NewGame

void UI::NewGame();

返回 无

描述 开始新游戏。首先调用单例 Score 的 newGame 方法，然后显示提示信息，调用 UI::GuessNumber() 进入猜数流程。循环调用直到该方法返回真表示猜对，显示祝贺信息。

UI::GuessNumber

bool UI::GuessNumber();

返回 是否猜对

描述 等待用户输入。判断用户的输入，如果是 8888 则调用 UI::ViewDetail()，如果是 7777 则调用 UI::ViewAnswer，如果是 0000 则返回上一级，如果是负数则直接退出程序。对于正常的输入，则直接调用 Score 实例的 guess() 方法，显示其返回的提示字符串，显示当前得分。

UI::ViewDetail

void UI::ViewDetail();

返回 无

描述 8888 功能。调用 Score 实例的 guess() 方法，并显示提示字符串。

UI::ViewAnswer

void UI::ViewAnswer();

返回 无

描述 7777 功能。首先调用 UI::InputPassword() 来进行密码输入和验证，允许三次密码输入，如果验证失败则直接返回；如果验证成功，则调用 Score 实例的 guess() 方法，并显示提示字符串。

UI::InputPassword

bool UI::InputPassword();

返回 密码是否验证通过

描述 提示用户输入密码，设置控制台属性，等待用户输入。在 Windows 下，密码输入后回显成星号 “*”；在 Linux 下，密码输入不回显，但退格键仍应可用。这些操作可以调用 getpass() 函数，在 Password.h 中提供。密码输入后，调用 Score 实例的 checkPassword() 方法来验证密码的正确性。

2.3 主要算法流程图

产生一个指定的随机数的 `Number::genRand()` 方法，即每次从 0..9 中选取一个数放到数组里。由于产生的是四位数，要求千位不能为 0，所以产生千位时采用了特殊处理。具体算法的伪代码描述见图 5。

```
NUMBER::GENRAND()
1  S = {}
2  T = {1, 2, ..., 9}
3  S[0] = 从 T 中随机选取一个数
4  T = T - {S0} + {0}
5  for i = 1 to 3
6      S[i] = 从 T 中随机选取一个数
7      T = T - {Si}
8  n =  $\overline{S_0 S_1 S_2 S_3}$ 
9  return n
```

图 5: 生成符合要求的随机数

验证猜数并返回指定的 (x, y) 的 `Number::guess()` 方法，首先需要从 `this→numbers` 获取答案的分解，然后把传进来的数字也进行分解。分解后的两个数组按位比较，累加后得 x 。两个数组求交集之后得到 $x + y$ 。具体算法见图 6。

验证猜数并返回详细信息的 `Number::detail()` 方法，和 `Number::guess()` 非常类似，只是这回返回的是一个数组。由于要求首位表示为 1，所以需要加一处理。具体算法伪代码见图 7。

语言细节相关的实现，不是前提设计的重点，这里不再赘述算法具体应该怎样用语言实现，请参考语言相关文档和标准库文档。

```

NUMBER::GUESS( $n$ )
1   $x = 0$ 
2   $y = 0$ 
3   $S = \text{this} \rightarrow \text{numbers}$ 
4   $\overline{A_0 A_1 A_2 A_3} = n$ 
5  for  $i = 0$  to 3
6      if  $A[i] == S[i]$ 
7           $x++$ 
8   $y = |S \cap A| - x$ 
9   $\text{this} \rightarrow \text{count}++$ 
10 return  $(x, y)$ 

```

图 6: 验证猜数并返回指定的 (x, y)

```

NUMBER::DETAIL( $n$ )
1   $V = \{\}$ 
2   $S = \text{this} \rightarrow \text{numbers}$ 
3   $\overline{A_0 A_1 A_2 A_3} = n$ 
4  for  $i = 0$  to 3
5      if  $A[i] == S[i]$ 
6           $V = V + \{i + 1\}$ 
7   $\text{this} \rightarrow \text{count}++$ 
8  return  $V$ 

```

图 7: 验证猜数并返回详细信息

3 程序清单及注释

4 运行结果与分析

5 总结

A 股票交易系统

这个附录是股票交易系统的设计文档。当初最开始的选题是难度系数为 1.2 的股票交易系统，着手设计了类和头文件之后，发现代码量有点大，不是同组成员可以一周之内可以完成的，于是抛弃了这个项目，改换了代码量较小的猜数字游戏。为了使劳动成果不至于白白浪费，就在这里把设计好的文档和代码一并放在课程设计的附录之中，以供参考。

A.1 设计类图

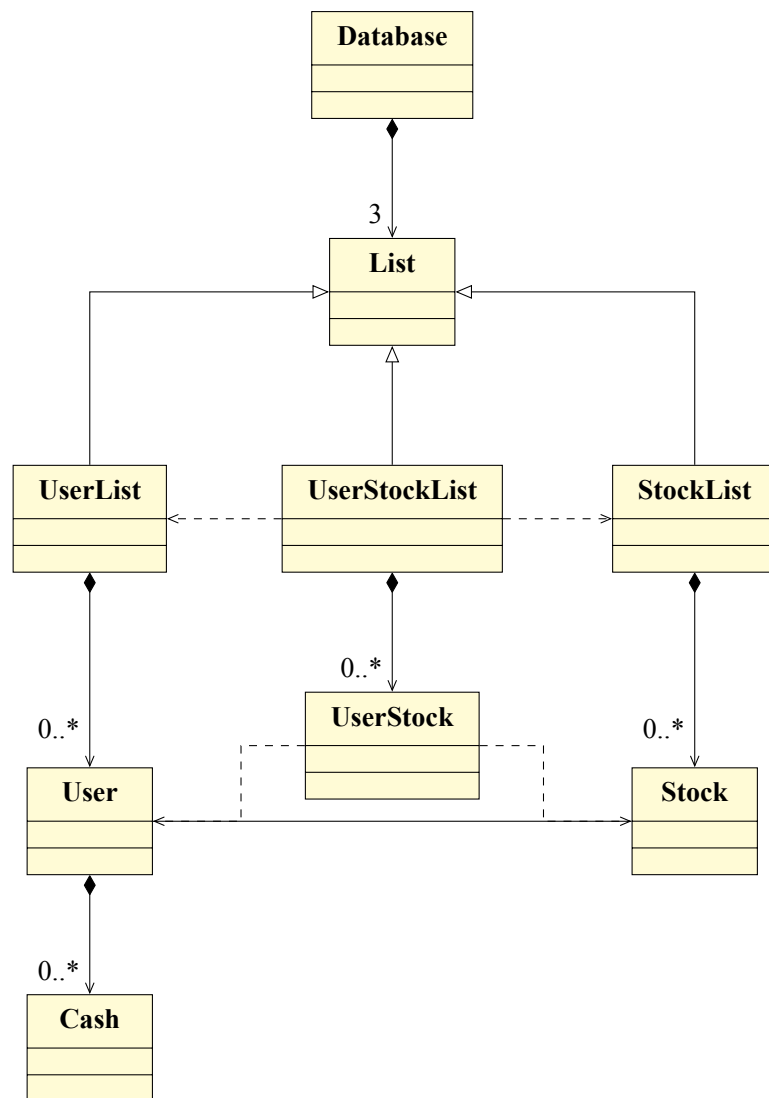


图 8: 股票交易系统各个类之间的关系

A.2 题目内容

(1) 修改数据结构，增加现金成员，每只股票增加牌价；每个用户的数据库中同样也增加现金数目的成员。

(2) 增加股票交易系统的接口程序，新增如下设计：

AddNewStock() 增加新股票

DeleteOldStock() 删除旧股票

HangUpStock() 挂起股票，停止交易

ModifyStock() 修改股票的名称、代码

以上修改均须输入密码，密码吻合后才能进入数据库进行修改，结果均存入 Stock_File.dat 中。

(3) 将股票数据的处理由数组改为链表，可以处理多只股票的交易，链表以交易代码的序号进行排序，也可根据需要对股票的牌价进行排序。(1.2)

A.3 头文件

```
1  /* Database.h
2  * 描述：数据库类，用于读取存储数据表
3  * 作者：田劲锋
4  * 创建时间：年月日2015111
5  * 修改时间：年月日2015111
6  */
7  #pragma once
8
9  #include <iostream>
10 #include <vector>
11 #include <string>
12 using namespace std;
13
14 #include "List.h"
15
16 class Database
17 {
18 public:
19     Database(string filename);
20     virtual ~Database();
21
22     void open();
23     void write();
24     void close();
25
26     List& getTable(string tablename) const;
27     List& getTable(int index) const;
28
29 protected:
30     string filename;
31     istream file;
32     vector<List> db;
33 };
```

```
1  /* List.h
2  * 描述：数据表类，用于描述数据库表
3  * 作者：田劲锋
4  * 创建时间：年月日2015111
5  * 修改时间：年月日2015111
6  */
7  #pragma once
8
9  #include <iostream>
10 #include <vector>
11 #include <string>
12 using namespace std;
13
14 class List
```



```

15 {
16 public:
17     List(string tablename);
18     virtual ~List();
19
20     void setTablenNme(string);
21     string getTableName() const;
22
23     int newId();
24
25     virtual int getTotal();
26
27 protected:
28     int max_id; // 标识表中最大的 ID
29     string tablename; // 表名
30     int total; // 表中元素数量
31 };

```

```

1  /* UserList.h
2  * 描述: 用户表类
3  * 作者: 田劲锋
4  * 创建时间: 年月日2015111
5  * 修改时间: 年月日2015111
6  */
7  #pragma once
8
9  #include <iostream>
10 #include <vector>
11 #include <string>
12 using namespace std;
13
14 #include "List.h"
15 #include "User.h"
16
17 class UserList : public List
18 {
19 public:
20     UserList(string tablename);
21     virtual ~UserList();
22
23     int getTotal();
24
25     User& getUser(int index) const;
26
27     User& findUser(string keyword, int startIndex = 0) const;
28     int findUserIndex(string keyword, int startIndex = 0) const;
29
30     int insert(const User&);
31     void remove(int index);
32     void remove(int startIndex, int number);
33
34     void sort(bool asc = true);
35
36 protected:
37     vector<User&> users;
38 };

```

```

1  /* StockList.h
2  * 描述: 股票表类
3  * 作者: 田劲锋
4  * 创建时间: 年月日2015111
5  * 修改时间: 年月日2015111
6  */
7  #pragma once
8
9  #include <iostream>
10 #include <vector>
11 #include <string>
12 using namespace std;
13
14 #include "List.h"
15 #include "Stock.h"
16
17 class StockList : public List
18 {
19 public:
20     StockList(string tablename);
21     virtual ~StockList();
22
23     int getTotal();
24
25     Stock& getStock(int index) const;
26
27     Stock& findStock(string keyword, int startIndex = 0) const;
28     int findStockIndex(string keyword, int startIndex = 0) const;
29

```

```

30     int insert(const Stock&);
31     void remove(int index);
32     void remove(int startIndex, int number);
33
34     void sort(bool asc = true);
35     void sortByCode(bool asc = true);
36     void sortByPrice(bool asc = true);
37     void sortByTotal(bool asc = true);
38
39 protected:
40     vector<Stock&> Stocks;
41 };

```

```

1  /* UserStockList.h
2  * 描述：用户股票表类
3  * 作者：田劲锋
4  * 创建时间：年月日2015111
5  * 修改时间：年月日2015111
6  */
7  #pragma once
8
9  #include <iostream>
10 #include <vector>
11 #include <string>
12 using namespace std;
13
14 #include "List.h"
15 #include "UserStock.h"
16
17 class UserStockList : public List
18 {
19 public:
20     UserStockList(string tablename);
21     virtual ~UserStockList();
22
23     int getTotal();
24
25     UserStock& getUserStock(int index) const;
26
27     UserStock& findUserStock(string keyword, int startIndex = 0) const;
28     int findUserStockIndex(string keyword, int startIndex = 0) const;
29
30     int insert(const UserStock&);
31     void remove(int index);
32     void remove(int startIndex, int number);
33
34     void sort(bool asc = true);
35
36 protected:
37     vector<UserStock&> us;
38 };

```

```

1  /* User.h
2  * 描述：用户类
3  * 作者：田劲锋
4  * 创建时间：年月日2015111
5  * 修改时间：年月日2015111
6  */
7  #pragma once
8
9  #include <vector>
10 #include <string>
11 using namespace std;
12
13 #include "Cash.h"
14 #include "Stock.h"
15 #include "UserStock.h"
16
17 class User
18 {
19     friend class Stock;
20     friend class UserStock;
21 public:
22     User();
23     virtual ~User();
24
25     void setUsername(string);
26     string getUsername() const;
27
28     void setPassword(string);
29     bool checkPassword(string) const;
30
31     void setAdmin(bool);
32     bool isAdmin() const;
33
34     bool operator==(const User&) const;

```

```

35     bool operator<(const User&) const;
36
37     User(const User&);
38     User& operator=(const User&);
39
40     friend ostream& operator<<(ostream&, const User&);
41     friend istream& operator>>(istream&, const User&);
42
43 protected:
44     int id; // 唯一标识元素的 ID
45     string username;
46     string password;
47     double money; // 用户手上的钱
48     bool admin; // 是否为管理员
49     vector<Cash> cashes; // 用户持有的股票
50     double amount; // 用户持有股票的总金额
51 };

```

```

1  /* Stock.h
2  * 描述: 股票类
3  * 作者: 田劲锋
4  * 创建时间: 年月日2015111
5  * 修改时间: 年月日2015111
6  */
7  #pragma once
8
9  #include <vector>
10 #include <string>
11 using namespace std;
12
13 #include "User.h"
14 #include "UserStock.h"
15
16 class Stock
17 {
18     friend class User;
19     friend class UserStock;
20 public:
21     Stock();
22     Stock(int code, double price, string name = "");
23     virtual ~Stock();
24
25     void setCode(int);
26     void setCode(string);
27     string getCode() const;
28
29     void setName(string);
30     string getName() const;
31
32     void setPrice(double);
33     double getPrice() const;
34
35     void addTotal(int);
36     int getTotal() const;
37
38     friend ostream& operator<<(ostream&, const Stock&);
39     friend istream& operator>>(istream&, const Stock&);
40
41 protected:
42     int id; // 唯一标识元素的 ID
43     string code; // 交易代码
44     string name;
45     bool valid; // 是否可交易 / 挂起
46     double price; // 股价
47     int total; // 股票数量
48     vector<User&> users; // 持有该股票的用户
49 };

```

```

1  /* UserStock.h
2  * 描述: 用户股票类, 存储
3  * 作者: 田劲锋
4  * 创建时间: 年月日2015111
5  * 修改时间: 年月日2015111
6  */
7  #pragma once
8
9  #include <vector>
10 #include <string>
11 using namespace std;
12
13 #include "User.h"
14 #include "Stock.h"
15
16 class UserStock
17 {
18     friend class User;

```

```

19     friend class Stock;
20 public:
21     UserStock(const User&, const Stock&, int num = 0);
22     virtual ~UserStock();
23
24     void set(const UserStock&);
25     UserStock& get() const;
26
27     bool operator==(const UserStock&) const;
28     bool operator<(const UserStock&) const;
29
30     UserStock& operator=(const UserStock&);
31
32     friend ostream& operator<<(ostream&, const UserStock&);
33     friend istream& operator>>(istream&, const UserStock&);
34
35 protected:
36     int id; // 唯一标识元素的 ID
37     int userId;
38     int stockId;
39     int num; // 股票数量
40     double price; // 购入价 / 卖出价 (负)
41     time_t timestamp; // 交易时间
42 private:
43     User& user;
44     Stock& stock;
45 };

```

```

1  /* Cash.h
2  * 描述: 现金类, 描述有几股该股票
3  * 作者: 田劲锋
4  * 创建时间: 年月日2015111
5  * 修改时间: 年月日2015111
6  */
7  #pragma once
8
9  #include "Stock.h"
10
11 class Cash
12 {
13 public:
14     Cash(const Stock&, int = 0);
15     virtual ~Cash();
16
17 protected:
18     Stock& stock;
19     int num; // 股数
20     double amount; // 总价值
21 };

```

```

1  /* UI.h
2  * 描述: 用户界面类, 用于用户交互
3  * 作者: 田劲锋
4  * 创建时间: 年月日2015111
5  * 修改时间: 年月日2015111
6  */
7  #pragma once
8
9  #include <vector>
10 #include <string>
11 using namespace std;
12
13 #include "Database.h"
14 #include "User.h"
15
16 class UI
17 {
18 public:
19     UI();
20     virtual ~UI();
21
22     void MainMenu();
23
24     void UserMenu();
25     void RegistUser(); // 注册新用户
26     void LoginUser(); // 用户登录
27
28     void StockMenu();
29     void ListStock(); // 显示股票列表
30     void ListStockByCode(); // 按交易代码顺序显示股票列表
31     void ListStockByPrice(); // 按价格顺序显示股票列表
32     void ListStockByTotal();
33     void BuyStock(); // 买入股票
34     void SaleStock(); // 卖出股票
35
36     void AdminMenu();

```

```
37     void AddNewStock(); // 增加新股票
38     void DeleteOldStock(); // 删除旧股票
39     void HangUpStock(); // 挂起股票, 停止交易
40     void ModifyStock(); // 修改股票的名称、代码
41
42 protected:
43     Database& db;
44     User& user;
45 };
```

```
1  #include <iostream>
2
3  #include "TestCase.h"
4
5  int main(int argc, char *argv[])
6  {
7      return 0;
8  }
```

参考文献

- [1] H. M. Deitel, P. J. Deitel. C++ 大学基础教程. 5 edn. 北京: 电子工业出版社, 2014
- [2] M. Gregoire, N. A. Solter, S. J. Kleper. C++ 高级编程. 2 edn. 北京: 清华大学出版社, 2012
- [3] I. Horton. Visual C++ 2012 入门经典. 6 edn. 北京: 清华大学出版社, 2013
- [4] 徐言生, (Editor) 设计模式解析. 2 edn. 北京: 人民邮电出版社, 2013
- [5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, et al. Introduction to Algorithms. 3 edn. London, England: The MIT Press, 2009
- [6] D. E. Knuth. The Art Of Computer Programming. Pearson Education, 1968–2011
- [7] 邓建松, 彭冉冉, 陈长松. L^AT_EX 2_ε 科技排版指南. 北京: 科学出版社, 2001