河南工業大學

课程设计报告

猜数字游戏

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　201316920315

王增辉　201316920309

任课教师：王献荣

完成时间：2015 年 1 月 15 日

目录

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。

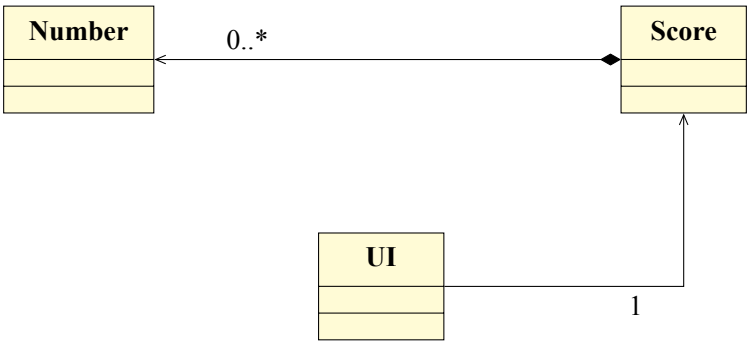

图 1: 猜数字游戏各个类之间的关系

2.2　类的设计说明

2.2.1　Number 类

Number
number: *int*　　//待猜的数 – numbers: *int[]*　//数的拆分数组 # count: *int*　　//猜数次数
+ Number()　　//初始化产生随机数 + setNumber(number: *int*): *void*　　//拆分 – 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 10 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，这两个是一对一的关系，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 11 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 11 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

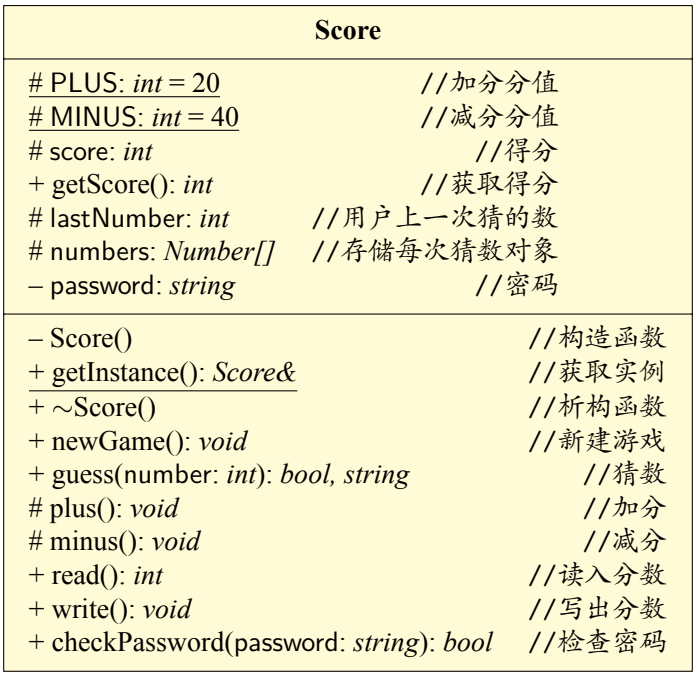

图 3: Score 类

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。初始化 this→lastNumber 为特殊值表示用户还没有输入任何数。

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::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”，注意如果 this→lastNumber 被标记成特殊值则表示用户直接输入了 8888 来作弊，这时候应该返回猜数失败和错误提示，而不应该调用 detail() 来返回帮助信息；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，则置 this→lastNumber 为 number，并调用 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。如果文件不存在，则初始化 this→score 为 0，初始化 this→password 为“root”，并调用 this→write() 方法将其写出。写出后再尝试读入。

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。要写入的文件名应该和 this→read() 中的读入文件保持一致，为了避免 Magic Number 是使用，这里应该使用一个全局的常量或者静态变量来存储。

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

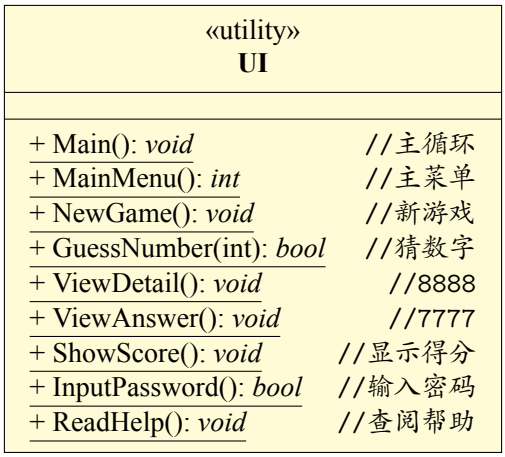

图 4: UI 类

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；(3) 显示帮助；(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，等待用户输入。判断用户的输入，如果是小于等于 0 的数则返回上一级表示结束本轮游戏；否则调用 UI::GuessNumber() 进入猜数流程，循环这个过程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 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()。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::ShowScore

```
void UI::ShowScore();
```

返回 无

描述 显示得分。

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。

UI::ReadHelp

```
void UI::ReadHelp();
```

返回　无

描述　显示如下帮助提示信息：

1. 游戏目的是猜一个四位数，且这个四位数每位都不相同
2. 每次猜数都会有提示

   分别表示数字、位置都匹配的个数，数字匹配但位置不匹配的个数
3. 猜对了加 20 分，猜错了减 40 分
4. 猜 8888 可以得到详细的提示
5. 猜 7777 可以直接看答案，但需要密码
6. 猜 0 或负数会退出该轮游戏，但仍会计分哦

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 $T = \{1, 2, \cdots, 9\}$
3 $S[0] =$从 T 中随机选取一个数
4 $T = T - \{S_0\} + \{0\}$
5 **for** $i = 1$ **to** 3
6 $\quad S[i] =$从 T 中随机选取一个数
7 $\quad T = T - \{S_i\}$
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 = this→numbers
4  A_0A_1A_2A_3 = n
5  for i = 0 to 3
6        if A[i] == S[i]
7              x++
8  y = |S ∩ A| − x
9  this→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 = this→numbers
3  A_0A_1A_2A_3 = n
4  for i = 0 to 3
5        if A[i] == S[i]
6              V = V + {i + 1}
7  this→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

3.1 许可证

3.2 工程文件

Listing 1: `Makefile`

```
RM=rm -f
CC= gcc
CXX= g++
DEFS=
PROGNAME= number
INCLUDES= -I.
LIBS=

ifdef WIN32
OSDEF= -fexec-charset=gbk
else
OSDEF= -DSYS_UNIX=1
endif

DEFINES= $(INCLUDES) $(DEFS) $(OSDEF)
CFLAGS= $(DEFINES)

SRCS = Number.cpp Score.cpp UI.cpp mylib.cpp cli.cpp

OBJS = Number.o Score.o UI.o mylib.o cli.o

.cpp.o:
        $(RM) $@
        $(CXX) $(CFLAGS) -c $*.cpp

all: $(PROGNAME)

$(PROGNAME) : $(OBJS)
        $(CXX) $(CFLAGS) -o $(PROGNAME) $(OBJS) $(LIBS)

clean:
        $(RM) $(OBJS) $(PROGNAME) core *~
```

Listing 2: `Makefile.vs`

```
CC= cl
CXX= cl
#DEFS= -nologo -DSTRICT -G3 -Ow -W3 -Zp -Tp
DEFS= -nologo /EHsc
PROGNAME= number.exe
LINKER=link -nologo

INCLUDES= -I.

DEFINES= $(INCLUDES) $(DEFS) -DWINNT=1 -DWIN32=1

CFLAGS= $(DEFINES)

SRCS = Number.cpp Score.cpp UI.cpp mylib.cpp cli.cpp

OBJS = Number.obj Score.obj UI.obj mylib.obj cli.obj

.cpp.obj:
        $(CXX) $(CFLAGS) -c $< -Fo$@

all: $(PROGNAME)

$(PROGNAME) : $(OBJS)
        $(LINKER) $(OBJS) /OUT:$(PROGNAME) $(LIBS)

clean:
        del $(OBJS) $(PROGNAME) core
```

3.3 头文件

Listing 3: `Number.h`

```
/* Number.h
* 描述: 猜数字类
* 作者: 田劲锋
* 创建时间: 2015-1-11
* 修改时间: 2015-1-15
*/
#pragma once

#include <vector>
#include <algorithm>

class Number
{
public:
    Number();
    virtual ~Number();

    void setNumber(int number);

    std::pair<int, int> guess(int);
    std::vector<int> detail(int);
    int answer();

protected:
    int number;
    int count; // 猜数次数
private:
    int numbers[4];
    int genRand();
};
```

Listing 4: `Score.h`

```
/* Score.h
* 描述: 得分类
* 作者: 田劲锋
* 创建时间: 2015-1-11
* 修改时间: 2015-1-15
*/
#pragma once

#include <vector>
#include <string>
#include <algorithm>

#include "Number.h"

class Score
```

```
{
    friend class Number;
public:
    virtual ~Score();

    static Score& getInstance();

    void newGame();

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

    int read();
    void write();

    int getScore() const;

    bool checkPassword(std::string password);

protected:
    int score;
    int lastNumber; // 用户上一次猜的数
    std::vector<Number> numbers;

    const static int PLUS = 20;
    const static int MINUS = 40;

    void plus();
    void minus();

private:
    Score();
    std::string password;
};
```

Listing 5: `UI.h`

```
/* UI.h
* 描述: 用户界面类, 用于用户交互
* 作者: 田劲锋
* 创建时间: 2015-1-11
* 修改时间: 2015-1-15
*/
#pragma once

#include "Score.h"

class UI
{
public:
    static void Main();

    static int MainMenu();

    static void NewGame();

    static bool GuessNumber(int n);

    static void ViewDetail();

    static void ViewAnswer();

    static void ShowScore();

    static bool InputPassword();

    static void ReadHelp();
};
```

3.4 主程序

Listing 6: `cli.cpp`

```
/* cli.cpp
* 描述: 猜数字游戏的命令行界面
* 作者: 田劲锋
* 创建时间: 2015-1-11
* 修改时间: 2015-1-12
*/
#include "UI.h"

int main(int argc, char *argv[])
```

```
{
    UI::Main();

    return 0;
}
```

3.5 邢志鹏的实现

3.6 田劲锋的实现

Listing 7: Number_tjf.cpp

```
/* Number_tjf.cpp
* 描述: 数字类的实现
* 作者: 田劲锋
* 创建时间: 2015-1-15
* 修改时间: 2015-1-15
*/
#include <cstdlib>
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

#include "Number.h"

Number::Number()
{
    this->setNumber(Number::genRand());
    this->count = 0;
}

Number::~Number()
{}

int Number::genRand()
{
    int s[4];
    vector<int> t;
    for (int i = 1; i <= 9; i++)
        t.push_back(i);
    int k = rand() % t.size();
    s[0] = t[k];
    t.erase(t.begin() + k);
    t.push_back(0);
    for (int i = 1; i <= 3; i++) {
        k = rand() % t.size();
        s[i] = t[k];
        t.erase(t.begin() + k);
    }
    int n = s[0] * 1000 + s[1] * 100 + s[2] * 10 + s[3];
    return n;
}

vector<int> itov(int n)
{
    vector<int> v(4);
    v[0] = n / 1000;
    v[1] = n % 1000 / 100;
    v[2] = n % 100 / 10;
    v[3] = n % 10;
    return v;
}

void Number::setNumber(int number)
{
    this->number = number;
    vector<int> v = itov(number);
    for (int i = 0; i < 4; i++) {
        this->numbers[i] = v[i];
    }
}

pair<int, int> Number::guess(int number)
{
    int x = 0, y = 0;
    vector<int> s = itov(this->number);
    vector<int> a = itov(number);
    int v[10] = { 0 };
    for (int i = 0; i < 4; i++) {
```

```
        x += (a[i] == s[i]);
        v[a[i]]++;
        v[s[i]]++;
    }
    for (int i = 0; i < 10; i++) {
        y += (v[i] == 2);
    }
    y = y - x;
    this->count++;
    return make_pair(x, y);
}

vector<int> Number::detail(int number)
{
    vector<int> v;
    int *s = this->numbers;
    vector<int> a = itov(number);
    for (int i = 0; i < 4; i++) {
        if (a[i] == s[i]) {
            v.push_back(i + 1);
        }
    }
    this->count++;
    return v;
}

int Number::answer()
{
    this->count++;
    return this->number;
}
```

Listing 8: `Score_tjf.cpp`

```
/* Score_tjf.cpp
* 描述: 得分类的实现
* 作者: 田劲锋
* 创建时间: 2015-1-15
* 修改时间: 2015-1-15
*/

#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <vector>
#include <algorithm>
#include <cstdlib>
#include <ctime>
using namespace std;

#include "Score.h"

int Score::getScore() const
{
    return this->score;
}

Score::Score()
{
    srand((unsigned int) time(NULL));
    this->score = 0;
    this->password = "root";
    this->lastNumber = -1;
    this->read();
}

Score& Score::getInstance()
{
    static Score instance;
    return instance;
}

Score::~Score()
{
    this->write();
}

void Score::newGame()
{
    Number n;
    this->numbers.push_back(n);
}

pair<bool, string> Score::guess(int number)
{
    Number &n = this->numbers.back();
```

```
    if (number == 8888) {
        if (this->lastNumber == -1) {
            return make_pair(false, "还没有猜过任何数");
        }
        vector<int> a = n.detail(this->lastNumber);
        ostringstream s;
        s << "第 ";
        for (size_t i = 0; i < a.size() - 1; i++) {
            s << (i + 1) << ",";
        }
        s << a.back() << "位数字正确";
        return make_pair(false, s.str());
    } else if (number == 7777) {
        int a = n.answer();
        ostringstream s;
        s << "正确答案是 " << a;
        return make_pair(false, s.str());
    } else {
        pair<int, int> a = n.guess(this->lastNumber = number);
        int x = a.first, y = a.second;
        ostringstream s;
        if (x == 4 && y == 0) {
            s << "猜中了，答案就是 " << number;
            plus();
            return make_pair(true, s.str());
        } else {
            s << "数位匹配 " << x << " 个，数匹配位不符 " << y << " 个";
            minus();
            return make_pair(false, s.str());
        }
    }
}

void Score::plus()
{
    this->score += Score::PLUS;
}

void Score::minus()
{
    this->score -= Score::MINUS;
}

const char* score_filename = "score.dat";

int Score::read()
{
    ifstream fin(score_filename);
    if (fin.good()) {
        fin >> this->password;
        fin >> this->score;
    } else {
        this->write();
    }
    return this->score;
}

void Score::write()
{
    ofstream fout(score_filename);
    if (fout.good()) {
        fout << this->password << endl;
        fout << this->score << endl;
    }
}

bool Score::checkPassword(string password)
{
    return password == this->password;
}
```

Listing 9: `UI_tjf.cpp`

```
/* UI_tjf.cpp
* 描述: 用户界面类的实现
* 作者: 田劲锋
* 创建时间: 2015-1-15
* 修改时间: 2015-1-15
*/

#include <iostream>
#include <cstdlib>
#include <string>
#include <algorithm>
using namespace std;

#include "UI.h"
```

```cpp
#include "mylib.h"

void UI::Main()
{
    int n = 1;
    while (n) {
        n = UI::MainMenu();
        if (n == 1) {
            UI::NewGame();
        } else if (n == 2) {
            UI::ShowScore();
        } else if (n == 3) {
            UI::ReadHelp();
        } else {
            break;
        }
        pause();
    }
}

int geti()
{
    cin.clear();
    string str;
    getline(cin, str);
    int n = atoi(str.c_str());
    return n;
}

int UI::MainMenu()
{
    cls();

    cout << endl;
    cout << " 猜数字游戏 " << endl;
    cout << "--------------" << endl;
    cout << "1) 新游戏 " << endl;
    cout << "2) 查成绩" << endl;
    cout << "3) 看帮助" << endl;
    cout << "0) 退出" << endl;
    cout << endl;
    cout << "---> ";

    int n = geti();
    return n;
}

void UI::ReadHelp()
{
    cout << endl
        << " 猜数字游戏帮助  " << endl
        << "--------------------" << endl
        << "1. 游戏目的是猜一个四位数，且这个四位数每位都不相同" << endl
        << "2. 每次猜数都会有提示" << endl
        << " 分别表示数字、位置都匹配的个数，数字匹配但位置不匹配的个数  " << endl
        << "3. 猜对了加 20 分，猜错了减 40 分" << endl
        << "4. 猜 8888 可以得到详细的提示" << endl
        << "5. 猜 7777 可以直接看答案，但需要密码" << endl
        << "6. 猜 0 或负数会退出该轮游戏，但仍会计分哦" << endl
        << endl
        << "**** 享受你的游戏！ ****" << endl;
}

void UI::NewGame()
{
    Score& s = Score::getInstance();
    s.newGame();
    bool flag = false;
    while (!flag) {
        cout << "请猜一个四位数: " << endl;
        cout << "--> ";
        int n = geti();
        if (n <= 0) {
            cout << "结束本轮游戏！ " << endl;
            return;
        }
        flag = UI::GuessNumber(n);
    }
    cout << "恭喜你猜对了！ " << endl;
}

bool UI::GuessNumber(int n)
{
    if (n == 8888) {
        UI::ViewDetail();
        return false;
    } else if (n == 7777) {
        UI::ViewAnswer();
        return false;
```

```
    }
    cout << "你猜的是: " << n << endl;
    Score& s = Score::getInstance();
    pair<bool, string> a = s.guess(n);
    cout << a.second << endl;
    UI::ShowScore();
    return a.first;
}

void UI::ViewDetail()
{
    Score& s = Score::getInstance();
    cout << s.guess(8888).second << endl;
}

void UI::ViewAnswer()
{
    for (int i = 0; i < 3; i++) {
        if (UI::InputPassword()) {
            Score& s = Score::getInstance();
            cout << s.guess(7777).second << endl;
            return;
        } else {
            cout << "密码错误! " << endl;
        }
    }
}

bool UI::InputPassword()
{
    Score& s = Score::getInstance();
    string t("请输入密码: ");
    string password(getpass((char *) t.c_str()));
    cout << endl;
    return s.checkPassword(password);
}

void UI::ShowScore()
{
    Score& s = Score::getInstance();
    cout << "当前得分: " << s.getScore() << endl;
}
```

3.7 辅助文件

这个是用 C 语言一些跨平台控制台控制的代码，提供在这里供调用。

Listing 10: `mylib.h`

```
/* mylib.h
* 描述: 提供一些跨平台的控制台控制函数
* 作者: 田劲锋
* 创建时间: 2015-1-12
* 修改时间: 2015-1-14
*/
#pragma once

char *getpass(char *prompt);
// 输入密码

void cls();
// 清屏

void pause();
// 暂停
```

Listing 11: `mylib.cpp`

```
/* mylib.cpp
* 描述: 提供密码的输入功能
* 作者: 田劲锋
* 创建时间: 2015-1-12
* 修改时间: 2015-1-15
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

```

```c
#include "mylib.h"

#ifdef WIN32
#include <conio.h>
#else
#include <termios.h>
#endif

#define PWD_MAX 256

char *getpass(char *prompt)
{
    static char passwd[PWD_MAX] = "";
    printf("%s", prompt);
#ifdef WIN32
    int i = 0;
    char c;
    while ((c = _getch()) != '\n' && c != '\r' && i < PWD_MAX) {
        if (c == '\b' && i > 0) {
            passwd[i--] = '\0';
            printf("\b \b");
        } else {
            passwd[i++] = c;
            printf("*");
        }
    }
    passwd[i] = '\0';
#else
    struct termios oldflags, newflags;
    tcgetattr(fileno(stdin), &oldflags);
    newflags = oldflags;
    newflags.c_lflag &= ~ECHO;
    newflags.c_lflag |= ECHONL;
    if (tcsetattr(fileno(stdin), TCSANOW, &newflags) != 0) {
        perror("tcsetattr");
        return NULL;
    }
    fgets(passwd, PWD_MAX, stdin);
    if (tcsetattr(fileno(stdin), TCSANOW, &oldflags) != 0) {
        perror("tcsetattr");
        return NULL;
    }
#endif
    return passwd;
}

void cls()
{
#ifdef WIN32
    system("cls");
#else
    system("clear");
#endif
}

void pause()
{
#ifdef WIN32
    system("pause");
#else
    system("read -p 请按任意键继续\". . .\"");
#endif
}
```

4　运行结果与分析

5 总结

贡献	权值	田劲锋	邢志鹏	王增辉
文档	50%	252行(39.6%)	384行(60.4%)	0行(0%)
代码	50%	行(%)	行(%)	行(%)
合计				

表 1: 小组贡献比例表

5.1 田劲锋的总结

5.2 邢志鹏的总结

5.3 王增辉的总结

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

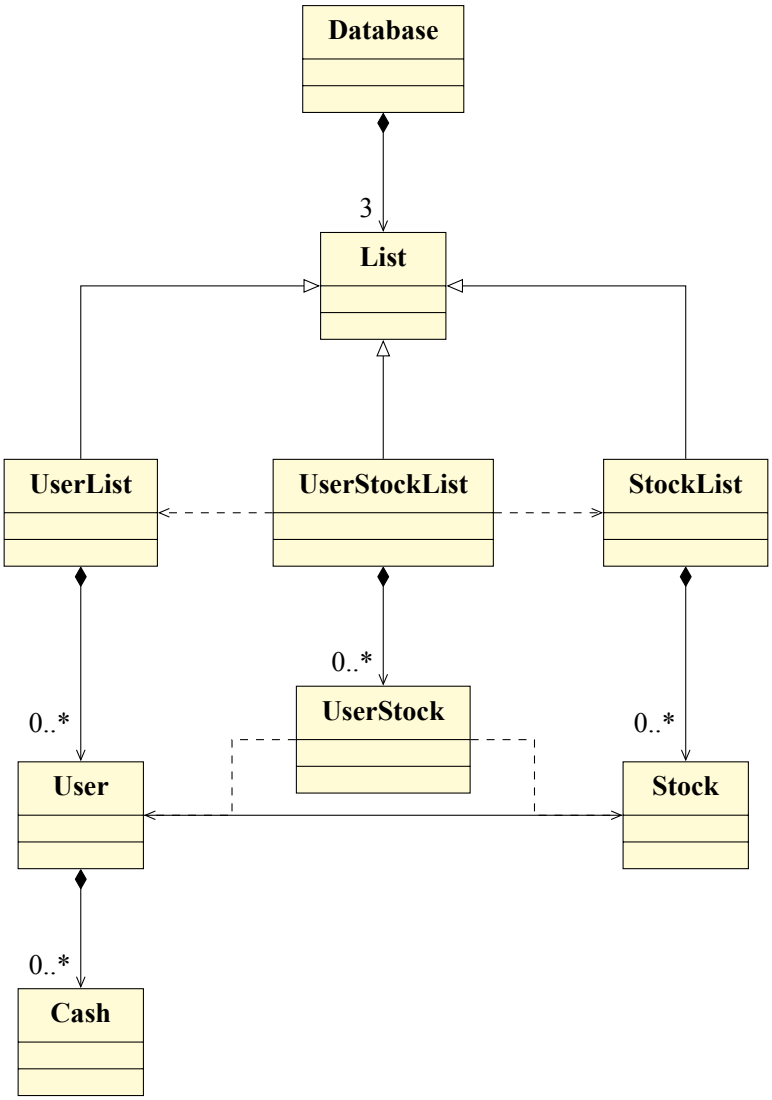

图 8: 股票交易系统各个类之间的关系

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，也可根据需要以股票的牌价进行排序。(1.2)

A.3 头文件

```
/* Datebase.h
 * 描述：数据库类，用于读取存储数据表
 * 作者：田劲锋
 * 创建时间：年月日2015111
 * 修改时间：年月日2015111
 */
#pragma once

#include <iostream>
#include <vector>
#include <string>
using namespace std;

#include "List.h"

class Database
{
public:
    Database(string filename);
    virtual ~Database();

    void open();
    void write();
    void close();

    List& getTable(string tablename) const;
    List& getTable(int index) const;

protected:
    string filename;
    iostream file;
    vector<List&> db;
};
```

```
/* List.h
 * 描述：数据表类，用于描述数据库表
 * 作者：田劲锋
 * 创建时间：年月日2015111
 * 修改时间：年月日2015111
 */
#pragma once

#include <iostream>
#include <vector>
#include <string>
using namespace std;

```

```
class List
{
public:
    List(string tablename);
    virtual ~List();

    void setTablenNme(string);
    string getTableName() const;

    int newId();

    virtual int getTotal();

protected:
    int max_id; // 标识表中最大的 ID
    string tablename; // 表名
    int total; // 表中元素数量
};
```

```
/* UserList.h
* 描述: 用户表类
* 作者: 田劲锋
* 创建时间: 年月日2015111
* 修改时间: 年月日2015111
*/
#pragma once

#include <iostream>
#include <vector>
#include <string>
using namespace std;

#include "List.h"
#include "User.h"

class UserList : public List
{
public:
    UserList(string tablename);
    virtual ~UserList();

    int getTotal();

    User& getUser(int index) const;

    User& findUser(string keyword, int startIndex = 0) const;
    int findUserIndex(string keyword, int startIndex = 0) const;

    int insert(const User&);
    void remove(int index);
    void remove(int startIndex, int number);

    void sort(bool asc = true);

protected:
    vector<User&> users;
};
```

```
/* StockList.h
* 描述: 股票表类
* 作者: 田劲锋
* 创建时间: 年月日2015111
* 修改时间: 年月日2015111
*/
#pragma once

#include <iostream>
#include <vector>
#include <string>
using namespace std;

#include "List.h"
#include "Stock.h"

class StockList : public List
{
public:
    StockList(string tablename);
    virtual ~StockList();

    int getTotal();

    Stock& getStock(int index) const;

    Stock& findStock(string keyword, int startIndex = 0) const;
    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
```

```
    bool operator==(const User&) const;
    bool operator<(const User&) const;

    User(const User&);
    User& operator=(const User&);

    friend ostream& operator<<(ostream&, const User&);
    friend istream& operator>>(istream&, const User&);

protected:
    int id; // 唯一标识元素的 ID
    string username;
    string password;
    double money; // 用户手上的钱
    bool admin; // 是否为管理员
    vector<Cash> cashes; // 用户持有的股票
    double amount; // 用户持有股票的总金额
};
```

```
/* Stock.h
 * 描述: 股票类
 * 作者: 田劲锋
 * 创建时间: 年月日2015111
 * 修改时间: 年月日2015111
 */
#pragma once

#include <vector>
#include <string>
using namespace std;

#include "User.h"
#include "UserStock.h"

class Stock
{
    friend class User;
    friend class UserStock;
public:
    Stock();
    Stock(int code, double price, string name = "");
    virtual ~Stock();

    void setCode(int);
    void setCode(string);
    string getCode() const;

    void setName(string);
    string getName() const;

    void setPrice(double);
    double getPrice() const;

    void addTotal(int);
    int getTotal() const;

    friend ostream& operator<<(ostream&, const Stock&);
    friend istream& operator>>(istream&, const Stock&);

protected:
    int id; // 唯一标识元素的 ID
    string code; // 交易代码
    string name;
    bool valid; // 是否可交易 / 挂起
    double price; // 股价
    int total; // 股票数量
    vector<User&> users; // 持有该股票的用户
};
```

```
/* UserStock.h
* 描述: 用户股票类, 存储
* 作者: 田劲锋
* 创建时间: 年月日2015111
* 修改时间: 年月日2015111
*/
#pragma once

#include <vector>
#include <string>
using namespace std;

#include "User.h"
#include "Stock.h"

class UserStock
{
```

```
    friend class User;
    friend class Stock;
public:
    UserStock(const User&, const Stock&, int num = 0);
    virtual ~UserStock();

    void set(const UserStock&);
    UserStock& get() const;

    bool operator==(const UserStock&) const;
    bool operator<(const UserStock&) const;

    UserStock& operator=(const UserStock&);

    friend ostream& operator<<(ostream&, const UserStock&);
    friend istream& operator>>(istream&, const UserStock&);

protected:
    int id; // 唯一标识元素的 ID
    int userId;
    int StockId;
    int num; // 股票数量
    double price; // 购入价 / 卖出价(负)
    time_t timestamp; // 交易时间
private:
    User& user;
    Stock& stock;
};
```

```
/* Cash.h
 * 描述: 现金类, 描述有几股该股票
 * 作者: 田劲锋
 * 创建时间: 年月日2015111
 * 修改时间: 年月日2015111
 */
#pragma once

#include "Stock.h"

class Cash
{
public:
    Cash(const Stock&, int = 0);
    virtual ~Cash();

protected:
    Stock& stock;
    int num; // 股数
    double amount; // 总价值
};
```

```
/* UI.h
* 描述: 用户界面类, 用于用户交互
* 作者: 田劲锋
* 创建时间: 年月日2015111
* 修改时间: 年月日2015111
*/
#pragma once

#include <vector>
#include <string>
using namespace std;

#include "Database.h"
#include "User.h"

class UI
{
public:
    UI();
    virtual ~UI();

    void MainMenu();

    void UserMenu();
    void RegistUser(); // 注册新用户
    void LoginUser(); // 用户登录

    void StockMenu();
    void ListStock(); // 显示股票列表
    void ListStockByCode(); // 按交易代码顺序显示股票列表
    void ListStockByPrice(); // 按价格顺序显示股票列表
    void ListStockByTotal();
    void BuyStock(); // 买入股票
    void SaleStock(); // 卖出股票

```

```
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] A. Shalloway, J. R. Trott. 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. LaTeX 2_ε科技排版指南. 北京: 科学出版社, 2001