

C++面向对象程序设计与实践-2

单词消除游戏系统程序说明书

班级: 2015211312

姓名: 祁瑞

学号: 2015211471

时间: 2017年5月

学院: 计算机学院

## 题目一

### 1.设计要求

* 实现闯关者，出题者本地的注册，登录。
* 程序支持多人注册，同一时间只有一人可以登录。
* 任何角色均可查询所有闯关者，出题者，按照属性查找相应闯关者，出题者。
* 可以根据闯关者闯过关卡数，经验，等级等对闯关者排名，根据出题者出题数目，等级对出题者排名。

### 2.设计思路

* 共设计了4个类，User-用户类 Player-玩家类 TestBuilder-出题者类 Game-游戏类。

**User**

**Player**

**TestBuilder**

类继承关系图

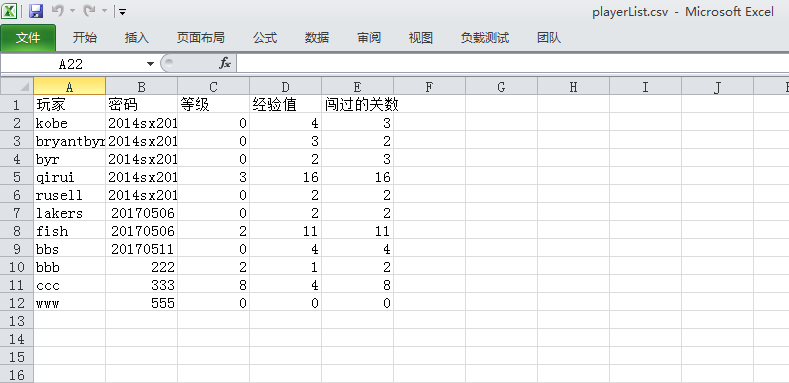
**Player**

**Game**

**TestBuilder**

类调用关系图

* 用户数据采用.csv文件的存储方式，存储格式对用户更加友好。有playerList.csv、testerList.csv、wordList.csv三个文件，分别存储挑战者信息、出题者信息、题库。



* 采用容器的数据存储方式，存储方式更加灵活，长度可自由伸缩。
* 为方便文件更新，自定义了字符分割数组函数

vector<string> split(const string& s, const string& delim);

### 3.关键算法流程图

出题者登录

注册

退出

玩家登录

选择游戏难度

开始游戏

下一关

查看排行榜

查询用户

退出

出题

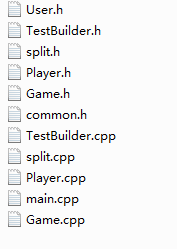
查看排行榜

查询用户

退出

开始

### 4.程序模块结构



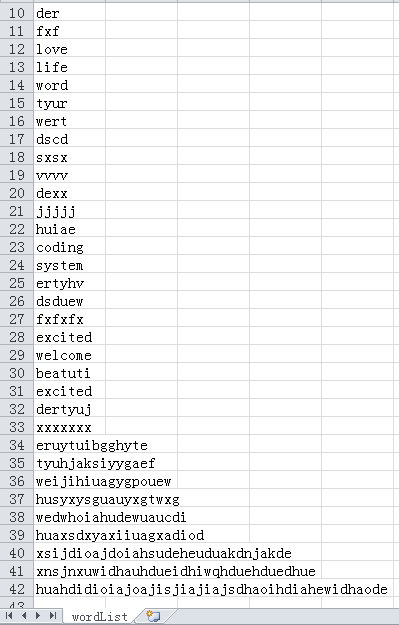
## 题目二

### 1.设计要求

* 必须在题目一的基础上进行修改。
* 请根据要求设计每一关的出题方式，注意随着关卡数增加，题目难度增加。请合理处理出题人新添加新词的使用方式，并且新加词组不会影响游戏难度。
* 设计闯关者经验值，等级增加策略。出题者等级升级策略。

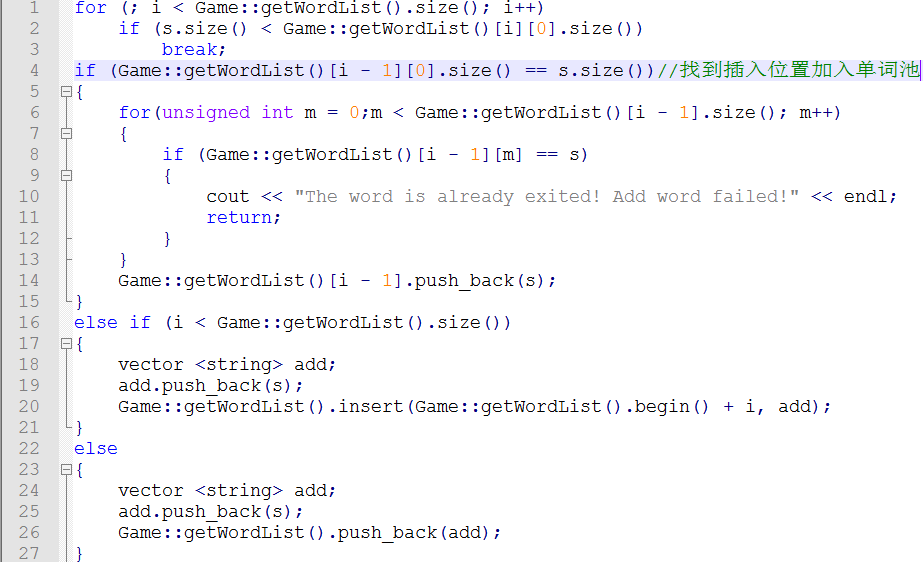
### 2.设计思路与实现

* 用二维vector容器 vector<vector> 组织单词，词库中的单词构成一个单词池，根据单词的长度来组织存储。每次出题时，系统从该单词池中按照关卡难度随机的选择相应长度的单词。随着关卡的升级，单词长度也越来越长。单词库效果图如下：



* 出题者新加单词不会影响游戏难度，出题者新增的单词会被自动识别难度系数并插入到单词池的合理位置。（此处设计存储文件更新）

**实现方法：**



* 闯关者及出题者升级策略为：



### 3.关键算法流程图

出题者登录

注册

退出

玩家登录

选择游戏难度

开始游戏

下一关

查看排行榜

查询用户

退出

出题

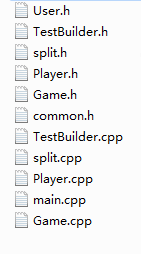
查看排行榜

查询用户

退出

开始

### 4.程序模块结构



## 题目三

### 1.设计要求

* 除单人游戏外，增加双人对战游戏，要求参与闯关者均已经登录，双人同时面对一个单词，最先打出正确单词者获得经验增长，在双人对战中获胜所获得的经验增长 要高于 从同等难度的单人游戏中所获得的经验增长，失败者则需要扣除一定经验值。
* 可以查看同时在线的游戏闯关者，可以挑战在线的游戏玩家，被挑战者接受挑战后进入双人对战。

### 2.实现思路与实现

1. 使用socket进行通信技术
2. 采用客户端/服务器（C/S）处理模型

a) 客户端主要完成界面交互功能，如接收用户输入数据、显示服务端处理结果等；

b) 服务端主要完成存储和计算等处理功能，如完成登录/注销操作、存取数据、处理

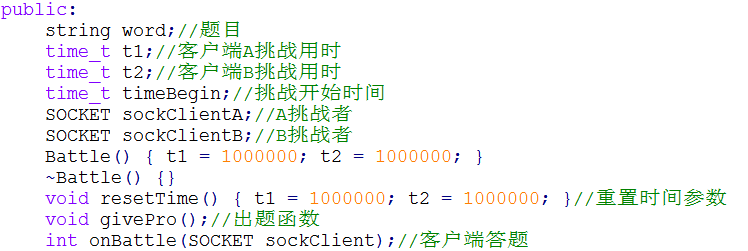
排序等，所有闯关者，出题者信息保存在服务器。

**Client**

**Sever**

1. 增加了对战类Battle, 用于实现双人对战。

关于实现双人对战的方式，涉及操作双线程的问题，由于两个客户端分别通过两个线程与服务器交互 ，为了实现对战的同步性，所以对战的信息应该以全局变量的形式存储。对战过程中两个玩家可以同步获取对战信息，否则，不能保证对战的同步性。为此，增加了双人对战类Battle, Battle类持有两个挑战者的socket等信息，以便于与两个用户进行及时通信。Battle类主要属性设计如下:



闯关者可**通过服务器端**向在线玩家发送消息邀其进行对战，对方接受后进入对战模式。

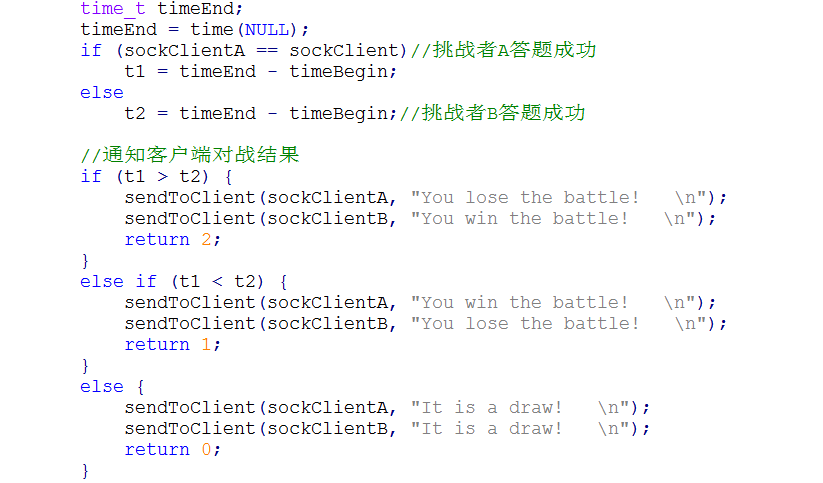
全局数据信息

**Sever**

**Client1**

**Client2**

对战过程中，服务器同时向两客户端发送数据，对战中及时更新存放在服务器端的用时等数据信息，通知对战对象对战结果。具体实现如下：

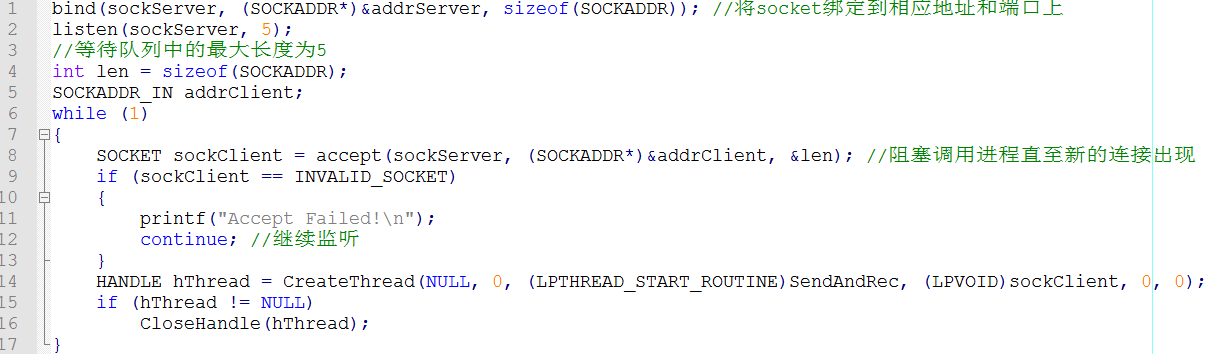


4、采用tcp多线程通信技术，服务器收到一个客户端请求，就给其分配一个线程与其通信，与此同时，记录该客户端端口号对应的闯关者。便于查看在线用户列表及进行双人对战。

//关联的socket

**实现方法**：

服务器收到一个客户端请求，就给其分配一个线程:



### 3.服务器端关键算法流程图

出题者登录

注册

退出

玩家登录

选择游戏难度

开始游戏

下一关

查看排行榜

查询用户

退出

出题

查看排行榜

查询用户

退出

SEVER –收到客户端请求

获取在线玩家列表

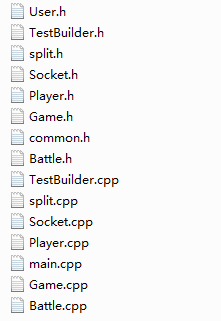
接受挑战

拒绝挑战

进入对战

邀请玩家进入对战战

### 4.程序模块结构



## 实验总结与感想

前两个题目实现过程相对顺利，没遇到太大的困难，主要的难点是我设计的文件的更新与定位操作。关于最后一个题目，自学了socket网络编程的部分内容，实现的最后一个题目的设计，在设计双人对战模式的时候，我想出了两个设计方案：

**方案一：两个挑战者分别在各自的线程中答题，由于的局部变量，最终各自的答题用时等信息必须发送给对方从而判定对战结果。**

**方案二：在服务器端设计全局变量用于存储两个挑战者各自用时等信息，服务器端通过全局变量可直接判定对战结果然后通知对战者。**

比较上述两个方案，在对战过程中，方案一仍然需要两个客户端进行通信; 而方案二不

需两个客户端再次进行通信。显然方案二的实现效率更高，且新增对战类使系统易于维护。于是我选择了方案二并加以实现。

实验中遇到的较大的Bug涉及文件更新操作，由于针对特定的某一行进行操作，重新写数据的时候可能会发生内存覆盖，所以一定要为每一行分配足够的缓存。

通过实验，再实践中进一步熟悉了C++语言的基本语法规则，掌握了基本的面向对象的程序设计方法及各种面向对象的程序设计技术。最重要的是体会到了掌握新技能的兴奋之感，感觉编程能力又有了一定程度地提高，并且对编程提高了兴趣。虽然编写代码的过程中会遇到各种棘手的问题，但解决问题之后的快感是令人振奋的。**问题会有的，解决问题的方法也会随之到来！**

## 程序使用说明

本程序的开发环境为：操作系统-Win7操作系统

集成开发环境-Visual Studio 2013

编译器-MSVC编译器

1、系统为便于管理用户，默认出题者的用户名以't'开头。出题者注册时，用户名需要以't'开头，其他情况下默认为玩家注册，而挑战者注册不可使用以't'开头的用户名，否则默认为出题者注册。

2、用户需正常退出，以保证系统信息的正常更新。