**单词消除游戏c++大作业程序说明**

姓名：张舒阳 学号：2015211519

**题目一**

**任务描述：**

单词消除游戏由两类参与者组成，闯关者（即游戏玩家），出题者（为游戏增加游戏中使用单词）。游戏规则为，游戏每一轮，程序会根据该关卡难度，显示一个单词，一定时间后单词消失。闯关者需要在相应地方输入刚刚显示并消失的单词，如果闯关者输入正确（即闯关者输入的单词与刚刚显示的单词完全一致，包含大小写）则为通过。一关可以由一轮或者多轮组成。

闯关者属性要求必须含有：闯关者姓名、已闯关关卡数、闯关者经验值、闯关者等级，若有需要可以自行添加其他属性。出题者属性要求必须含有：出题者姓名、出题者出题数目，等级，若有需要可以自行添加其余属性。

具体功能包括：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。

**开发环境：**

cfree5

**数据结构：**

User基类：

class user{

public:

string name;**//姓名**

string ID;**//ID号**

user();**//构造函数**

};

Player：//挑战者小类

class player:public user**//公有继承user**

{

private:**//私有属性**

int pass;**//已闯关卡数**

int exp;**//经验值**

int level;**//等级**

public:**//公有属性**

player\*nextptr；**//指向下一个player**

player()**//构造函数**

int flag;**//排序时需要用到的标志位**

int getpass();**//得到最高过关数**

int getexp();**//得到经验值**

int getlevel();**//得到等级**

void writepass(int a);**//写最高过关数**

void writeexp(int a);**//写经验值**

void writelevel(int a);/**/写等级**

void compute\_exp\_and\_level(int n);**//游戏结束后用到的计算经验值和等级的函数**

};

Designer：//出题者小类

class designer:public user**//公有继承user**

{

private:**//私有属性**

int num;**//出题数目**

int level;**//等级**

public:**//共有属性**

designer\*nextptr;**//指向下一个出题者**

int flag; **//排序时用到的标记位**

designer();**//构造函数**

int getnum();**//得到出题数**

int getlevel()**;//得到等级**

void writenum(int a)**;//写出题数**

void writelevel(int a);**//写等级**

void compute\_level();**//计算等级**

};

Word：**//单词小类**

class word

{

public: **//共有属性**

string W;**//单词**

int order;**//单词序号**

word\*nextptr;**//指向下一个词的指针**

word();**//构造函数**

};

Wordlist：**//单词大类**

class wordlist

{

public:

word \*headptr,\*curptr;**//指向单词列表的第一个词和最后一个词**

bool addaword();**//添加一个单词**

void saveinfile();**//存入文件**

void init\_wordlist();**//对单词表初始化，从文件当中读取**

word\*search(string w);**//传入单词返回指向单词的指针**

string getword(int i);**//通过序号找到单词**

int game();**//游戏函数**

wordlist(); **//构造函数**

};

Playerlist: **//玩家大类**

class playerlist

{

public:**//公有属性**

player\*headptr,\*curptr;**//指向第一个玩家和最后一个玩家**

playerlist()**;//构造函数**

void initplayerlist();**//对玩家列表初始化**

void saveinfile();**//存入文件**

bool createplayer();**//创建一个新的玩家**

player\*simple\_search(string I);**//传入ID返回指向该玩家的指针**

void specific\_search(int a);**//根据不同的属性，去找相应的玩家**

void print\_player(string I);**//将该玩家信息打印出来**

void rank\_player();**//给玩家排序的函数，经验值越多等级越高**

void flagtozero();**//标记位归零函数**

void find\_player\_from\_exp(int exp);**//通过经验值打印玩家的信息**

};

Designerlist：//大类

class designerlist

{

public:

designer\*headptr,\*curptr;//**指向下一个出题者**

designerlist()**;//构造函数**

void initdesigner();**//初始化出题人列表**

void saveinfile();**//存入文件**

bool createdesigner();**//新建一个出题者，注册**

designer\*simple\_search(string I);**//遍历函数**

void specific\_search(int a);**//根据不同的属性 去找相应的出题者**

void print\_designer(string I);**//将该出题者信息打印出来**

void rank\_designer();**//给出题者排序的函数，出题越多等级越高**

void flagtozero();**//标记位归零函数**

void find\_designer\_from\_num(int num);**//通过num打印出题人的信息**

};

**类说明：**

Player，designer，word小类都是基本的存储单元，用来存放基本信息，并设置接口函数对其中的私有属性进行修改。

Playerlist，designerlist，wordlist大类用来管理所有的玩家，出题者，以及单词。通过函数实现对小类的添加，修改，初始化，存入文件，排序，打印等操作。

**实现功能：**

1.玩家或出题者登录

可以提示：姓名与ID不符，登陆成功，您还没有注册。

2.玩家或出题者注册

可以提示：ID已被占用，注册成功。

1. 注销

回到登录与注册的界面

1. 玩家登录成功后可以完成的操作

A 查排名

玩家（根据经验值排名）

相关信息有：姓名 ID 等级 最高通关数 经验值

出题者（根据出题数排名）

相关信息有：姓名 ID 等级 出题数

B 查自己的信息

ID 姓名 过关数 经验值 等级

C 按属性查玩家

可根据ID，姓名，最高过关数，经验值或者等级查询。

D 按属性查出题者

可根据ID，姓名，出题数，经验值或者等级查询。

E 单人游戏

a 游戏过程中：

每关3个词 每次记忆时间5秒

b 经验值增加策略

每关增加50经验值

d 等级增加策路

每升一级需要的经验值是100

1. 出题者登录成功后可完成的操作

A 出题

a 策略 每出十个题升一级

B 查排名

玩家（根据经验值排名）

相关信息有：姓名 ID 等级 最高通关数 经验值

出题者（根据出题数排名）

相关信息有：姓名 ID 等级 出题数

C 查自己的信息

姓名 ID 出题数 等级

D 按属性查玩家

可根据ID，姓名，最高过关数，经验值或者等级查询。

E 按属性查出题者

可根据ID，姓名，出题数，经验值或者等级查询。

**题目二**

**任务描述**

闯关者即为游戏玩家，已经注册并登录的玩家可以在系统进行单词消除游戏。每一关的难度要有所增加，体现为如下三个条件中的一个或者多个：1、单词难度可以递增或者持平（即长度加长或不变）； 2、进行轮数增多（即单词数目如：前三关仅仅通过一个单词就过关，后续需要通过两个，三个甚至更多才过关）； 3、单词显示时间缩短（随着关卡的增加显示时间越来越短）。

闯关者每闯过一关，增加一定经验值。经验值会根据闯过的该关卡的关卡号、该关的闯关耗费时间共同决定。当经验值累计到一定程度闯关者等级增加。闯关失败需要重新闯该关。

游戏自带词库，而且已经注册的出题者可以为系统出题，即增加词库的新词，已经存在的单词不能再次添加（词库中的单词构成一个单词池，但建议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）。每成功出题一次，更新该出题者的出题数目。出题者等级根据出题人成功出题数目来升级。

**开发环境：**

Cfree5

**数据结构：**

User基类：

class user{

public:

string name;**//姓名**

string ID;**//ID号**

user();**//构造函数**

};

Player：//挑战者小类

class player:public user**//公有继承user**

{

private:**//私有属性**

int pass;**//已闯关卡数**

int exp;**//经验值**

int level;**//等级**

public:**//公有属性**

player\*nextptr；**//指向下一个player**

player()**//构造函数**

int flag;**//排序时需要用到的标志位**

int getpass();**//得到最高过关数**

int getexp();**//得到经验值**

int getlevel();**//得到等级**

void writepass(int a);**//写最高过关数**

void writeexp(int a);**//写经验值**

void writelevel(int a);/**/写等级**

void compute\_exp\_and\_level(int n);**//游戏结束后用到的计算经验值和等级的函数**

};

Designer：//出题者小类

class designer:public user**//公有继承user**

{

private:**//私有属性**

int num;**//出题数目**

int level;**//等级**

public:**//共有属性**

designer\*nextptr;**//指向下一个出题者**

int flag; **//排序时用到的标记位**

designer();**//构造函数**

int getnum();**//得到出题数**

int getlevel()**;//得到等级**

void writenum(int a)**;//写出题数**

void writelevel(int a);**//写等级**

void compute\_level();**//计算等级**

};

Word：**//单词小类**

class word

{

public: **//共有属性**

string W;**//单词**

int order;**//单词序号**

word\*nextptr;**//指向下一个词的指针**

word();**//构造函数**

};

Wordlist：**//单词大类**

class wordlist

{

public:

word \*headptr,\*curptr;**//指向单词列表的第一个词和最后一个词**

bool addaword(string s);**//添加一个词**

void saveinfile(string filenum);**//存入文件**

void init\_wordlist(string filenum);**//根据不同的文件初始化不同难度的单词表**

word\*search(string w);**//传入单词返回指向该单词的指针**

string getword(int i);**//通过序号找到单词**

int game(wordlist low,wordlist mid,wordlist high);**//游戏函数**

wordlist(); **//构造函数**

};

Playerlist: **//玩家大类**

class playerlist

{

public:**//公有属性**

player\*headptr,\*curptr;**//指向第一个玩家和最后一个玩家**

playerlist()**;//构造函数**

void initplayerlist();**//对玩家列表初始化**

void saveinfile();**//存入文件**

bool createplayer();**//创建一个新的玩家**

player\*simple\_search(string I);**//传入ID返回指向该玩家的指针**

void specific\_search(int a);**//根据不同的属性，去找相应的玩家**

void print\_player(string I);**//将该玩家信息打印出来**

void rank\_player();**//给玩家排序的函数，经验值越多等级越高**

void flagtozero();**//标记位归零函数**

void find\_player\_from\_exp(int exp);**//通过经验值打印玩家的信息**

};

Designerlist：**//出题者大类**

class designerlist

{

public:

designer\*headptr,\*curptr;//**指向下一个出题者**

designerlist()**;//构造函数**

void initdesigner();**//初始化出题人列表**

void saveinfile();**//存入文件**

bool createdesigner();**//新建一个出题者，注册**

designer\*simple\_search(string I);**//遍历函数**

void specific\_search(int a);**//根据不同的属性 去找相应的出题者**

void print\_designer(string I);**//将该出题者信息打印出来**

void rank\_designer();**//给出题者排序的函数，出题越多等级越高**

void flagtozero();**//标记位归零函数**

void find\_designer\_from\_num(int num);**//通过num打印出题人的信息**

};

**类说明：**

Player，designer，word小类都是基本的存储单元，用来存放基本信息，并设置接口函数对其中的私有属性进行修改。

Playerlist，designerlist，wordlist大类用来管理所有的玩家，出题者，以及单词。通过函数实现对小类的添加，修改，初始化，存入文件，排序，打印等操作。

与题目一的不同之处在于，题目二游戏过程中单词难度改变，因此在wordlist大类中的初始化函数与题目一的有所不同，在题目二的main函数中，需要初始化三个难度不同的wordlist对象，游戏函数也需要传入这三个对象，在关数增加的时候改变使用的单词表。

**实现功能：**

0.可以检测输入时，输入注册1和登录2以外的错

1.登录

可以提示：姓名与ID不符，登陆成功，您还没有注册

2.注册

可以提示：此ID已被占用，注册成功

3.注销

回到登录与注册的界面

4.玩家登陆成功后可进行的操作

A 查排名

玩家（根据经验值排名）

相关信息有：姓名 ID 等级 最高通关数 经验值

出题者（根据出题数排名）

相关信息有：姓名 ID 等级 出题数

B 查自己的信息

ID 姓名 过关数 经验值 等级

C 按属性查玩家

可根据ID，姓名，最高过关数，经验值或者等级查询。

D 按属性查出题者

可根据ID，姓名，出题数，经验值或者等级查询。

E 单人游戏

a 游戏过程中：

第一关 共2轮（即2个词） 记忆时间5秒。

之后，随关数的增加，轮数每关增加一轮，记忆时间缩短1秒（减至2秒后不 再减短）。若输入错误，游戏结束。

词库选择： 1~2关,用初级词库 长度<=4

3~4关,用中级词库 5<=长度<=9

5关以后都用高级词库 10<=长度

b 游戏结束后向玩家回馈：

共写对单词数

所通关数

答题所用总时间（秒）

c 经验值增加策略

duration为所用总时间，lp为所通关数

对于玩家来说，每通一关经验值增加50，即lp\*50，但还要减去所用的总时间 duration（罚时），但是要注意，这里的通关数和玩家信息当中的最高通关数是不 同的。

d 等级增加策路

每升一级需要的经验值是100。

1. 出题者登录成功后可进行操作

A 出题

a 策略 每出十个题升一级

b 出题过程中

已存在 出题失败

长度<=4 初级词库

5<=长度<=9 中级词库

长度>=10 高级词库

B 查排名

玩家（根据经验值排名）

相关信息有：姓名 ID 等级 最高通关数 经验值

出题者（根据出题数排名）

相关信息有：姓名 ID 等级 出题数

C 查自己的信息

姓名 ID 出题数 等级

D 按属性查玩家

可根据ID，姓名，最高过关数，经验值或者等级查询。

E 按属性查出题者

可根据ID，姓名，出题数，经验值或者等级查询。

**题目三**

**任务描述**

在题目二的基础上，将游戏由本地单机，扩展为服务器多人游戏平台，使用客户端/服务器的方式。**将所有闯关者，出题者信息保存在服务器**。

* 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。
* 使用socket进行通信技术
* 需要完成服务器端程序，以及客户端程序，客户端可以启动多个同时与服务器交互，要求训练服务器并发处理能力。

**开发环境**

Cfree5

**数据结构**

User基类：

class user{

public:

string name;**//姓名**

string ID;**//ID号**

user();**//构造函数**

};

Player：//挑战者小类

class player:public user**//公有继承user**

{

private:**//私有属性**

int pass;**//已闯关卡数**

int exp;**//经验值**

int level;**//等级**

public:**//公有属性**

player\*nextptr；**//指向下一个player**

player()**//构造函数**

int logon; **//是否在线标志位**

int waitmatch;**//是否等待被挑战标志位**

string bechallenge; **//这位玩家被谁挑战，就把谁的ID号保存到这个串**

int flag;**//排序时需要用到的标志位**

int getpass();**//得到最高过关数**

int getexp();**//得到经验值**

int getlevel();**//得到等级**

void writepass(int a);**//写最高过关数**

void writeexp(int a);**//写经验值**

void writelevel(int a);/**/写等级**

void compute\_exp\_and\_level(int n);**//游戏结束后用到的计算经验值和等级的函数**

};

Designer：//出题者小类

class designer:public user**//公有继承user**

{

private:**//私有属性**

int num;**//出题数目**

int level;**//等级**

public:**//共有属性**

designer\*nextptr;**//指向下一个出题者**

int flag; **//排序时用到的标记位**

designer();**//构造函数**

int getnum();**//得到出题数**

int getlevel()**;//得到等级**

void writenum(int a)**;//写出题数**

void writelevel(int a);**//写等级**

void compute\_level();**//计算等级**

};

Word：**//单词小类**

class word

{

public: **//共有属性**

string W;**//单词**

int order;**//单词序号**

word\*nextptr;**//指向下一个词的指针**

word();**//构造函数**

};

Wordlist：**//单词大类**

class wordlist

{

public:

word \*headptr,\*curptr;**//指向单词列表的第一个词和最后一个词**

bool addaword(string s);**//添加一个词**

void saveinfile(string filenum);**//存入文件**

void init\_wordlist(string filenum);**//根据不同的文件初始化不同难度的单词表**

word\*search(string w);**//传入单词返回指向该单词的指针**

string getword(int i);**//通过序号找到单词**

int game(wordlist low,wordlist mid,wordlist high);**//游戏函数**

wordlist(); **//构造函数**

};

Playerlist: **//玩家大类**

class playerlist

{

public:**//公有属性**

player\*headptr,\*curptr;**//指向第一个玩家和最后一个玩家**

playerlist()**;//构造函数**

void initplayerlist();**//对玩家列表初始化**

void saveinfile();**//存入文件**

void check\_logon();**//检测有哪些玩家在线**

int check\_waitmatch();**//检测有哪些玩家等待对手的挑战**

bool createplayer();**//创建一个新的玩家**

player\*simple\_search(string I);**//传入ID返回指向该玩家的指针**

void specific\_search(int a);**//根据不同的属性，去找相应的玩家**

void print\_player(string I);**//将该玩家信息打印出来**

void rank\_player();**//给玩家排序的函数，经验值越多等级越高**

void flagtozero();**//标记位归零函数**

void find\_player\_from\_exp(int exp);**//通过经验值打印玩家的信息**

};

Designerlist：**//出题者大类**

class designerlist

{

public:

designer\*headptr,\*curptr;//**指向下一个出题者**

designerlist()**;//构造函数**

void initdesigner();**//初始化出题人列表**

void saveinfile();**//存入文件**

bool createdesigner();**//新建一个出题者，注册**

designer\*simple\_search(string I);**//遍历函数**

void specific\_search(int a);**//根据不同的属性 去找相应的出题者**

void print\_designer(string I);**//将该出题者信息打印出来**

void rank\_designer();**//给出题者排序的函数，出题越多等级越高**

void flagtozero();**//标记位归零函数**

void find\_designer\_from\_num(int num);**//通过num打印出题人的信息**

};

**类说明：**

Player，designer，word小类都是基本的存储单元，用来存放基本信息，并设置接口函数对其中的私有属性进行修改。

Playerlist，designerlist，wordlist大类用来管理所有的玩家，出题者，以及单词。通过函数实现对小类的添加，修改，初始化，存入文件，排序，打印等操作。

表面上看起来与题目二的差别并不大，但其实，大类中的每一个函数基本上都做了不同程度的修改。

因为要进行socket通信，当客户端将需求发来时，服务器分析了需求之后，就要将客户端需要的信息发回去。这个时候，为了方便发送，我将这些信息打包，在题目二中，直接打印到命令行的信息，此时在题目三中，我将其打印到一个打包文件里，然后，再让缓冲区数组将这个文件整体读入，发送到客户端，客户端要做的只是将缓冲区的内容打印出来。

另外，因为加入了查阅是否在线以及挑战功能，player类中的变量也增加了logon是否登录标志位和waitmatch是否等待被挑战标志位。

**Socket通信：**

服务端：

1. 创建套服务器套接字(socket)。  
   2.将套接字绑定到一个本地地址和端口上(bind)。  
   3.将套接字设为监听模式，准备接受客户请求(listen)。  
   4.等待客户请求，接受连接请求，返回一个新对应于此次连接的套接字(accept)。  
   5.用返回的套接字和客户端进行通讯(send/recv)。

6.关闭套接字。

客户端：

1. 创建服务器套接字(socket)。  
    2.向服务器发出连接请求(connect)。  
    3.和服务器端进行通讯(send/recv)。  
    4.关闭套接字。

在服务器main函数当中，初始化各个需要用到的对象，创建多个线程，每个线程函数都可以与一个客户端实现通信，由此实现服务器的并发处理能力。

**实现功能：**

1.登录

可以提示：

姓名与ID不符

登陆成功

您还没有注册

2.注册

可以提示：

此ID已被占用

注册成功

3.注销

回到登录与注册的界面

4.player

1.登录

可以提示：姓名与ID不符，登陆成功，您还没有注册

2.注册

可以提示：此ID已被占用，注册成功

3.注销

回到登录与注册的界面

4.玩家登陆成功后可进行的操作

A 查排名

玩家（根据经验值排名）

相关信息有：姓名 ID 等级 最高通关数 经验值

出题者（根据出题数排名）

相关信息有：姓名 ID 等级 出题数

B 查自己的信息

ID 姓名 过关数 经验值 等级

C 按属性查玩家

可根据ID，姓名，最高过关数，经验值或者等级查询。

D 按属性查出题者

可根据ID，姓名，出题数，经验值或者等级查询。

E 单人游戏

a 游戏过程中：

第一关 共2轮（即2个词） 记忆时间5秒。

之后，随关数的增加，轮数每关增加一轮，记忆时间缩短1秒（减至2秒后不 再减短）。若输入错误，游戏结束。

词库选择： 1~2关,用初级词库 长度<=4

3~4关,用中级词库 5<=长度<=9

5关以后都用高级词库 10<=长度

b 游戏结束后向玩家回馈：

共写对单词数

所通关数

答题所用总时间（秒）

c 经验值增加策略

duration为所用总时间，lp为所通关数

对于玩家来说，每通一关经验值增加50，即lp\*50，但还要减去所用的总时间 duration（罚时），但是要注意，这里的通关数和玩家信息当中的最高通关数是不 同的。

d 等级增加策路

每升一级需要的经验值是100。

F 双人游戏

a 游戏过程中：

每关一轮，每轮记忆时间均为3秒

b 每轮游戏结束后返回：

对方优先或是我方优先

如果出现错误则返回 对方错误或是我方错误

出现错误，游戏结束

c 经验值增加策略

优先过一关加60经验值

落后过一关扣5经验值

d 等级增加策路

每升一级需要的经验值是100

G 挑战功能

选择双人游戏后，出现挑战别人或等待被别人挑战的选项

选择前者 如果此时等待挑战的人数是0，则挑战失败，提示玩家稍后再来

如果此时有玩家等待被挑战，则显示这些玩家的基本信息，挑战者 查看后，挑选一位自己想挑战的玩家，并输入该玩家的ID号，此 时系统就将两者匹配成功，并在被挑战的玩家界面中，显示挑战他 的这位挑战者的基本信息，三秒后开始双人游戏。

选择后者 则该玩家将一直处于等待之中 直到有挑战者选择他作为 对手， 此时界面上显示挑战者的个人信息，三秒后开始双人游戏。

F 查看当前有哪些在线玩家，以及他们的基本信息

1. 出题者登录成功后可进行操作

A 出题

a 策略 每出十个题升一级

b 出题过程中

已存在 出题失败

长度<=4 初级词库

5<=长度<=9 中级词库

长度>=10 高级词库

B 查排名

玩家（根据经验值排名）

相关信息有：姓名 ID 等级 最高通关数 经验值

出题者（根据出题数排名）

相关信息有：姓名 ID 等级 出题数

C 查自己的信息（可输入检错）

姓名 ID 出题数 等级

D 按属性查玩家（可输入检错）

可根据ID，姓名，最高过关数，经验值或者等级查询。

E 按属性查出题者

可根据ID，姓名，出题数，经验值或者等级查询。

**实验总结：**

在题目一当中，我认为最需要注意的问题就是数据结构的建立。合理的数据结构为后来功能的扩充和完善打下了好的基础。但是我的程序也存在一个不小的问题，在完成较简单的题目一时，我将所有的内容都写在了一个cpp文件中实现，之后题目二，题目三的扩充使得一个cpp文件中的代码量变得非常大，影响到了代码的可读性，也让调试变得非常不便。

题目二相比题目一，功能上更加的合理和完善，关数增加，每轮的记忆时间减少，每轮的词数增加，每关单词的难度也在变化，同时出题者所出的单词也由长度的不同存在了不同的文件里。

题目三相比题目二，涉及到了socket通信。按要求，要加入联机对战，并且所有的数据都需要存储到服务器端。一开始觉得，以我小白的基础根本无法实现，但通过网络和学长学姐的帮助，自学了socket的Windows编程技术之后，对这部分的了解稍微深入了一些。其实socket通信就是一个服务器与客户端先建立联系，之后，服务器不断地分析客户端的发送来的需求，返回客户端想要的数据的过程。很重要的一点是，一收一发，必须同步，否则一定会在传输数据中出现错误。并且我自己设计了服务器发送数据之前，先在一个文件中打包数据的过程，使得传输的信息更加完备，传输过程也更加便捷。