

国家网络安全学院

C语言程序设计大作业报告

课程题目：**2048**升级版

成员及其分工：

曹成成：game、game2函数编写、 GAME函数编写

李筱婷：IfWin函数编写、添加音效、 GAME函数编写

刘一经：Move、GAME函数编写

郑洋：GameContiute函数、隐藏光 标、设置句柄

夏诗怡：添加文件、GamePrint函数

目录：

1. C语言课程大作业 问题定义、限制 要求————3
2. 创新亮点——————————————————4
3. 总流程图——————————————————5
4. 集成开发环境————————————————6
5. 主要问题及解决思路—————————————6
6. 主要程序段的说明—————————————7-29
7. 运行结果截图及解释———————————29-33

八、总结感想—————————————————33-35

一：c语言课程设计大作业

1. 题目：2048游戏升级版

2. 问题定义：2048游戏是一款数字益智游戏，初始数字是由2+2组成的基数4。在4X4的界面里通过移动数字获得分数并实现终极目标在16格范围中凑出[2048]数字方块

3. 限制：

在操作方面表现为：

（1）游戏刚开始会产生**两个**随机数字2或者4。

（2）用户每次可以选择上下左右其中一个方向去移动，（3）每滑动一次，所有的数字方块都会往滑动的方向 靠拢，数字相撞时相同的数字会**合并叠加。**

同时系统会在空白的地方随机出现一个数字：2or4。

（4）当界面填满数字，且无法在通过移动叠加相同数字，游戏失败。

4. 输出要求：获得2048数字方块取得胜利。

二、创新亮点

在原有的游戏规则上，通过团队线上及线下讨论交流；在思维火花的碰撞下，我们增加了以下元素：

优化游戏操作方面：

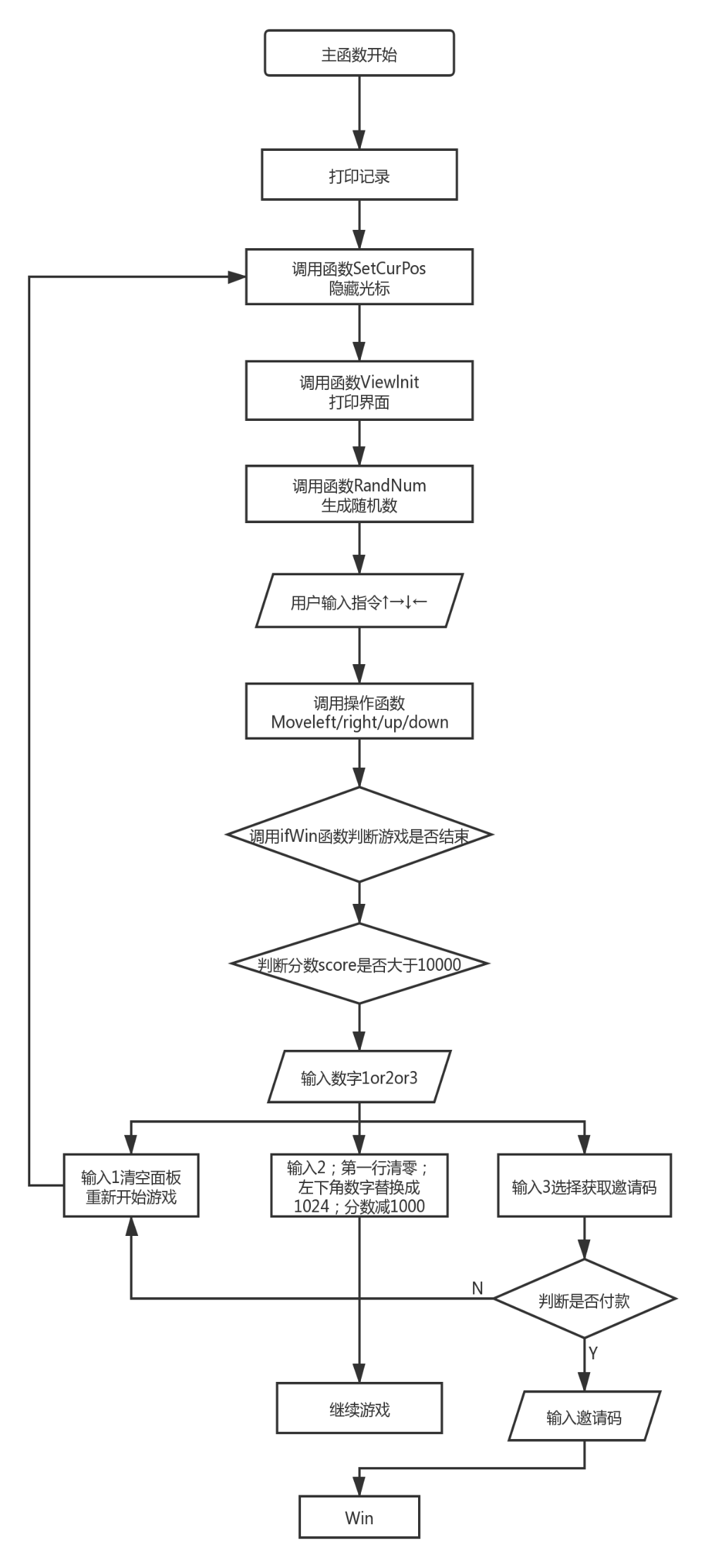
1. 记录游戏的最高分和游戏次数。
2. 添加音效：移动、碰撞叠加、失败、胜利、操作无法进行
3. 结束游戏后，可重新开始游戏，不用退出游戏界面。

增加游戏趣味方面：

4．当分数大于3000且游戏失败后，可选择清空第一行数字，并在左下角数字改为1024，从而继续游戏。（游戏经验表明当把最大数字放到某一角，更容易获得胜利）

5.分数大于10000，可选择通过联系客服得到邀请码直接获得胜利。（钞能力）

三、总流程图：



四、集成开发环境：c环境

五、主要问题的解决思路

主要问题：

1、初始化随机生成两个数字

2、打印表格

3、上下左右移动叠加及生成新的数字并计算成绩

4、记录历史成绩

5、额外设计的格子塞满后可以选择复活的良心机制。

解决思路

（**具体详细说明**在**六**中会介绍，一下均为**大致思路**）

1：利用srand((unsigned int)time(&t)获系统时间作为种子值，配合rand()随机数函数来产生一个随机的(x,y)的坐标，并给4\*4的数组中这个坐标的数赋值为2，重复两次。

2、打印表格：利用for循环的三重嵌套打印表格和随机生成的两个2。

3、上下左右移动：通过二维数组变换实现

4、历史成绩：添加文件将记录保存在record

5、复活：

在原有基础上，第一行数字赋值0，并给arr[3][0]赋值1024，后续进行另一个函数game2。

7、添加音效：将音频文件保存在D盘

六、主要程序段的说明

生成随机数：

\*源程序代码：

#include <time.h>

void RandNum(int row, int col) //生成随机数

{

int x = 0;

int y = 0;

do

{

x = rand() % 4;

y = rand() % 4;

} while (arr[x][y]);

arr[x][y] = (rand() % 10 ? 2 : 4);

}

void game()

{

srand((unsigned int)time(NULL));

在程序设计中，生成随机数要用到随机值函数rand（）

首先要包含头文件：#include<stdlib.h>。

rand()函数是按指定的顺序来产生整数，因此每次执行上面的语句都打印相同的两个值，所以说C语言的随机并不是真正意义上的随机，有时候也叫伪随机数。

使用 rand() 生成随机数之前需要用随机发生器的初始化函数 srand(unsigned seed)（也位于 stdlib.h 中） 进行伪随机数序列初始化，seed 又叫随机种子，通俗讲就是，如果每次提供的 seed 是一样的话，最后每一轮生成的几个随机值也都是一样的，因此叫伪随机数。

所以需要每次提供不同的 seed 达到完全的随机，我们通常用时间函数 time(NULL) 作为 seed ，因为时间值每秒都不同，这个函数需要包含头文件：#include<time.h>

time(NULL)的结果是一个类似于1524655706的数字，每秒都在递增1，从而达到srand的seed不断变化，不断生成新的随机数。

**拓展**

如果rand（）括号内不加参数，直接调用函数rand的话可能会生成下面这样的数：24810,14119,26009,20105等 所以我们想要生成指定范围内的随机数的话就需要用到求余运算符%，这里有一个规律，如果我们想要0-10的随机数时，就写成rand（）%11,0-100就写成rand（）&101，就是运算符后的数字需要比需求范围值大1（其实也就是取余运算的原理）

程序是4X4的界面，通过数列对每个空白赋值。所以arr[x][[y]中x和y都应该是0-3，所以对3取余。又因为要求生成4的概率为1/10。所以通过要求生成一个10以内的数字或者对10取余，判断是否为4，如果是4就生成4，不是就生成2。通过三叶运算符rand（）%10？2:4完成要求。

**Game2函数**

\*源程序

void game2()

{

ViewInit(20, 10);

GamePrint(ROW, COL);

while (1)

{

if (IfWin(ROW, COL) == 1)

{

printf("YOU WIN!\n");

PlaySound("D:\\胜利.wav", NULL, SND\_FILENAME | SND\_ASYNC);

reminder(score);

return;

}

else if (IfWin(ROW, COL) == -1)

{

printf("GAME OVER\n");

PlaySound("D:\\失败.wav", NULL, SND\_FILENAME | SND\_ASYNC);

reminder(score);

return;

}

else

{

if (GameContinue())

{

system("cls");

RandNum(ROW, COL);

GamePrint(ROW, COL);

}

else continue;

}

}

}

Game2（）函数是在game（）函数基础上编写出来的

在源程序中，当分数超过10000分时可选择消除分数从而消除第一行数字，并在左下角生成1024。由于game函数中包含了随机值初始化。所以在进行程序设计时，我选择了添加game2函数，保留了ViewInit、GamePrint、Move类函数，从而进行后续操作。

**int IfWin(int row, int col)**

{

int b = 0;//此处的b没什么意义，只是为了配合Move函数调用参数

int copy\_board[ROW][COL], i, j;

for (i = 0; i < ROW; i++)

{

for (j = 0; j < COL; j++)

{

copy\_board[i][j] = arr[i][j];

}

}//把原数组复制一遍

for (i = 0; i < row; i++)

for (j = 0; j < col; j++)

if (arr[i][j] == 2048)

return 1;//获胜

for (i = 0; i < row; i++)

for (j = 0; j < col; j++)

if (MoveLeft(copy\_board, &b) == 1 || MoveRight(copy\_board, &b) == 1 || MoveUp(copy\_board, &b) == 1 || MoveDown(copy\_board, &b) == 1)

return 0;//判断是否能移动

return -1;//游戏结束

}

解释：这个函数主要用来判断是否获胜或失败，当获胜（即数组中出现2048）时返回值是1，当失败时返回值是-1。同时这个函数还可用于判断移动是否可行，当该方向上的数不可合并时，不能移动，返回值在GameContinue中被调用。

问题：最初的程序没有判断移动是否可行就直接生成新随机数（根据原游戏规则，应当在执行移动后才能生成新随机数）。

解决方案：添加了一段判断代码，为了不改变原数组的值，引入copy\_board新数组把原数组复制了一遍，并让其试着运行Move函数，只要四个Move其中之一的返回值为1（即可以运行），就表示可以移动。

**音效**

（头文件中的一行）

#pragma comment(lib,"winmm.lib")

（GameContinue中的一段）

if (h == 0)

PlaySound("D:\\蜂鸣.wav", NULL, SND\_FILENAME | SND\_ASYNC);

else if (h == 1 && b == 0)

PlaySound("D:\\旋转.wav", NULL, SND\_FILENAME | SND\_ASYNC);

else

PlaySound("D:\\高级.wav", NULL, SND\_FILENAME | SND\_ASYNC);

return h;

（game函数中的一段）

if (IfWin(ROW, COL) == 1)

{

printf("YOU WIN!\n");

PlaySound("D:\\胜利.wav", NULL, SND\_FILENAME | SND\_ASYNC);

reminder(score);

return;

}

else if (IfWin(ROW, COL) == -1)

{

printf("GAME OVER\n");

PlaySound("D:\\失败.wav", NULL, SND\_FILENAME | SND\_ASYNC);

reminder(score);

return;

}

解释：为了使每一类操作有不同的音效，插入了五段不同的音频。单纯的滑动——“旋转”音效；碰撞产生累加效果——“高级”音效；移动无法执行——“蜂鸣”音效；游戏获胜——“胜利”音效；游戏失败——“失败音效”共五种。

问题：在适合的地方插入音频。获胜和失败的音频很好插入，直接在game函数输出胜负时播放即可，较为复杂的是游戏过程中的三种音效。最直接的方法是放在Move函数对应的操作步骤下，但是一开始Move函数里只有一个形参（m），而且该形参的返回值是为IfWin函数提供判断是否能移动的。

解决方案：一开始的解决方案是让两种操作（累加和单纯的滑动）的m返回值不同，但是由于Move函数的内部是一个循环结构，后一次的循环会遮盖前一次的值。而我们的预期目的是只要出现累加就播放“高级”音频，所以想到再添加一个参数记录累加操作是否完成，但是函数只能有一个返回值，m的返回值又不能省略，于是最后的解决方案是在Move函数中添加了一个**指针形参**，这样一来只需要在GameContinue添加一个实参，指针形参取它的地址赋值，即可完成多值传递。

实验报告之文件和GamePrint

**文件部分**

**目的**：通过创建文件实现游戏分数的保存以及最高纪录的显示。

**用于在文件中记录分数以及游戏结束时间的函数（源程序代码）：**

void reminder(int a)

{

time\_t time\_seconds = time(0);

//定义用于保存时间的变量；time()函数用于返回从1970年1月1日00：00：00至今经过的秒数

struct tm timer;//<time.h>头文件中定义的结构体

localtime\_s(&timer, &time\_seconds);//获取系统时间

FILE\* fp;//定义文件指针

errno\_t err;

//定义整形数据，用以返回错误代码（如果成功就返回0，失败则返回相应的非零错误代码）

err = fopen\_s(&fp, "records.txt", "a");

//用追加模式创建或打开文件

if (err)

{

printf("在记录数据的时候打开失败！\n");

exit(0);

}

else

{

fprintf\_s(fp, "%-8d%.2d.%.2d.%.2d:%.2d:%.2d\n",

a,

timer.tm\_year + 1900,

timer.tm\_mon + 1,

timer.tm\_mday,

timer.tm\_hour,

timer.tm\_min,

timer.tm\_sec);

fclose(fp);

}//向创建成功的文件中写入分数及游戏完成时间（若创建或打开文件失败，则打印出相应提示并退出）

}

**用于从文件中读取历史最高纪录并输出的函数（源程序代码）：**

int best(void)

{

char m = '0';

int a = 0, b = 0, max = 0, i = 0;

FILE\* fp;

errno\_t err;

err = fopen\_s(&fp, "records.txt", "a");

fclose(fp);//以追加模式创建或打开文件

err = fopen\_s(&fp, "records.txt", "r");

//用只读模式打开已存在的文件（若打开文件失败，则打印出相应提示并退出）

if (err)

{

printf("在打印历史记录的时候打开失败！\n");

exit(0);

}

else

{

fseek(fp, 0, 2);//重置文件位置指示至文件尾

k = ftell(fp) / 24;

//读取流变量k的当前位置号（已位于文件尾），并除以24以计算游戏完成次数

printf("您共有%d条游戏记录\n", k);

fseek(fp, 0, 0);重置文件位置指示至文件头

for (i = 0; i < k; i++)

{

fseek(fp, i \* 24, 0);//重置文件位置指示

fscanf\_s(fp, "%d", &a);//读取游戏分数

if (max < a)

max = a;

while (m != '\n')

m = fgetc(fp);

m++;

//将文件中记录时间的数据赋值给m，用于移动文件指示位置

fscanf\_s(fp, "%d", &b);

if (max < b)

max = b;

}//逐条读取游戏分数并比较大小

}

fclose(fp);

return max;//返回最高纪录

}

**思路历程：**

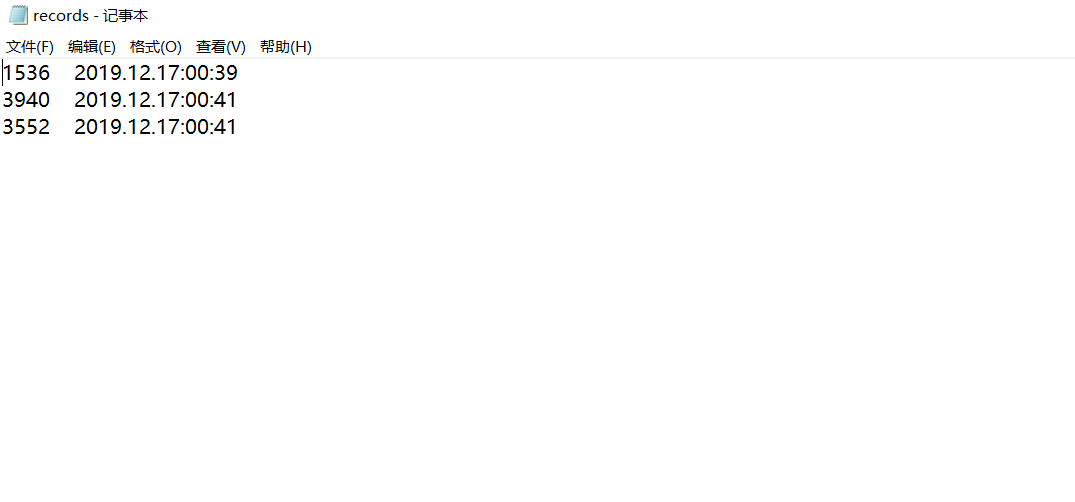
在组内成员讨论后，我们决定通过文件形式实现显示游戏记录这一功能，在每次游戏后记录分数和时间并展示给用户。

最初设想显示所有记录，程序简单但效果不佳（记录过多时显示界面难以设计），继而想到在文件中将分数排序后输出前三条，但改动文件中已有内容难度较大且没有必要，最后决定读取文件中存储的分数并比较大小、输出最大值，并增加提示语句，提示用户在文件中查看所有记录。编写过程中出现过许多错误，其中最难解决的是文件位置指示问题，难以保证变量a、b的赋值为游戏分数，最终解决方案是计算每一条记录的字符数量，并利用变量k确定游戏次数，移动文件指示位置，利用字符型变量m读取表示时间的字符来保证分数赋值。

**游戏记录显示效果展示**



**文件查看效果展示**



**显示及排版部分**

**目的：**输出令人愉悦的显示界面，提升游戏体验。

**用于制作显示界面的函数（源程序代码）：**

void GamePrint(int row, int col)

{

SetCurPos(5, 9);

printf("SCORE = %d", score);//设置光标位置并打印分数

SetCurPos(5, 10);

printf(" Highest=%d", best());

printf("\n您可以在您电脑中名为“records”的文件中\n所有的游戏记录\n");

//设置光标位置并打印最高纪录、查看文件提示

SetCurPos(48, 5);

int i = 0;

for (i = 0; i < row; i++)

{

for (int k = 0; k < 4; k++)

{

printf(" ");

for (int j = 0; j < 4; j++)

{

printf("-");

}

}

printf("\n\t\t\t\t\t\t");//用于移动光标至指定位置

for (int k = 0; k < 4; k++)

{

printf("|");

if (arr[i][k])

{

printf("%4d", arr[i][k]);//将数字填入表格

}

else

{

for (int j = 0; j < 4; j++)

{

printf(" ");//没有数字的位置用空格替代

}

}

}

printf("|\n\t\t\t\t\t\t");//用于移动光标至指定位置

if (i == row - 1)

{

for (int k = 0; k < 4; k++)

{

printf(" ");

for (int j = 0; j < 4; j++)

{

printf("-");

}

}//打印不在“— |”循环范围内的第9行

printf("\n\t\t\t\t\t\t");//用于移动光标至指定位置

}

}//使用循环语句打印表格并将游戏数字置入表格空位

SetCurPos(48, 4);

printf("上下左右表示方向\n");

//设置光标位置以打印游戏操作要求

SetCurPos(48, 15);//设置光标位置

}

**思路历程：**

2048游戏呈现的效果是数字在表格中的移动和变化，如果没有表格，游戏将枯燥无味，在参考、学习他人的设计后，我们利用循环语句、条件语句实现了表格的打印，虽然表格样式有些简陋，但游戏过程中的视觉体验已差强人意。

循环结构的设置从一次循环改为三次循环（用条件语句实现最后一行的打印），减少了代码数量。也在学习光标设置函数后将游戏主界面移动到了

屏幕的中间部分，优化设计。

**显示界面2.0版效果展示**



**显示界面最终版效果展示**



设置光标：

void SetCurPos(int x, int y) //设置光标的位置

{

HANDLE hStdOutput = GetStdHandle(STD\_OUTPUT\_HANDLE);//调用句柄

COORD coord = { x, y };

SetConsoleCursorPosition(hStdOutput, coord);//设置光标位置

}

HANDLE（句柄）是Windows操作系统中的一个概念。

HANDLE hStdOutput = GetStdHandle((STD\_OUTPUT\_HANDLE))在C语言中表示：从一个特定的标准设备（标准输出）中取得一个句柄（用来标识不同设备的数值）。其中HANDLE是定义句柄的申明，hStdOutput是句柄变量名，GetStdHandle是一个Windows API函数，在头文件Windows.h里。它用于从一个特定的标准设备（标准输入（STD\_INPUT\_HANDLE）、标准输出（STD\_OUTPUT\_HANDLE）或标准错误（STD\_ERROR\_HANDLE））中取得一个句柄（用来标识不同设备的数值）。最后的STD\_OUTPUT\_HANDLE就用来表示标准输出的句柄。

COORD是Windows API中定义的一种结构，表示一个字符在控制台屏幕上的坐标。其定义为：

typedef struct \_COORD {

SHORT X; // horizontal coordinate

SHORT Y; // vertical coordinate

} COORD;

由这个定义可知COORD这个结构体里存的就是一个二维坐标。

SetConsoleCursorPosition是Windows API中定位光标位置的函数。这里的作用就是把光标定位到输入的坐标coord上。

隐藏光标：

void ViewInit(int width, int height) //隐藏光标

{

HANDLE hStdOutput = GetStdHandle(STD\_OUTPUT\_HANDLE);

CONSOLE\_CURSOR\_INFO info;//声明一个结构体

GetConsoleCursorInfo(hStdOutput, &info);//检索光标

info.bVisible = 0;//使光标不可见

SetConsoleCursorInfo(hStdOutput, &info);//将修改值返回控制台

}

CONSOLE\_CURSOR\_INFO是包含控制台光标的信息的结构体。其定义如下：

typedef struct \_CONSOLE\_CURSOR\_INFO {

DWORD dwSize;// 光标百分比厚度（1~100），DWORD 表示 32bit 无符号整数

BOOL bVisible;// 是否可见，BOOL是布鲁型变量，0为FALSE，1为TRUE

} CONSOLE\_CURSOR\_INFO;

由此可见这个结构包含一个光标厚度参数和控制是否可见的参数。

GetConsoleCursorInfo和SetConsoleCursorInfo分别是检索和设置有关指定的控制台屏幕缓冲区的光标的可见性和大小信息。这里GetConsoleCursorInfo是检索hStdOutput（标准输出）的光标并把值赋给结构体info，然后设置光标不可见后利用SetConsoleCursorInfo函数将修改值返回控制台，从而起到了隐藏光标的作用。

GameContinue函数：

int GameContinue()

{

int ch1 = 0;

int ch2 = 0;

int h = 0, b = 0;

int (\*f[4])(int[4][4]) = { MoveUp,MoveDown,MoveLeft,MoveRight };

while (1)

{

if (ch1 = \_getch())

{

ch2 = \_getch();

if (ch2 == 72)

{

h = (\*f[0])(arr, &b);

break;

}

if (ch2 == 80)

{

h = (\*f[1])(arr, &b);

break;

}

if (ch2 == 75)

{

h = (\*f[2])(arr, &b);

break;

}

if (ch2 == 77)

{

h = (\*f[3])(arr, &b);

break;

}

}

}

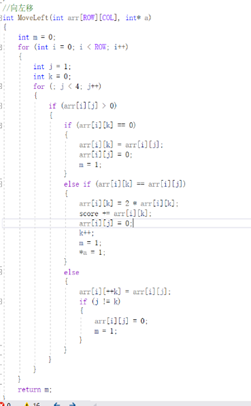
}

这里的GameContinue函数中主要的问题是如何获取↑↓←→键的ASCII码，搜集资料后我们发现↑↓←→键由一个扫描码和一个ASCII码组成，而函数getchar()不会读出扫描码，只能读出ASCII码，C语言中函数\_getch()【conio.h，不回显】可以读出扫描码。这里方向键的扫描码均是224，所以不具有选择性，而ASCII值则是，↑：72，↓：80，←：75，→：77。所以这里调用了两次\_getch()函数，第一次读扫描码，第二次读对应的ASCII码值，再根据ASCII值来判断并调用对应函数。调用函数时我们做了一些改良，用函数指针数组来更方便的调用函数。另外，move函数的返回值h是用于之后的声音和判断游戏是否能继续下去的。

一、Move 系列函数：MoveLeft函数、MoveRight函数、MoveUp函数、MoveDown 函数；

举例讲解：MoveLeft函数：

源代码截图：



想象每个数字即对应着是一个小格子，4行4列中的16个数字可以用一个二维数组来表示，我们在程序最初时设定了一个全局二维数组int arr[4][4],该二维数组arr[4][4]中的值即为游戏中4行4列中相对应的数值。 上下左右的移动即代表着对游戏进行对应的操作。

对于整个4行4列的数据的移动可以先将第一行中数值进行移动，最后用for循环将4行数据一行行的进行移动。

游戏规则有一条是：向某一方向上移动时，当4行中的每一行都无法执行移动或合并的操作时对该方向进行移动操作时无任何变化。为实现该功能，我组决定将Move系列函数设置为int 返回值类型的函数，返回一个整数0或1，0代表未对4行进行移动操作，1代表至少4行中有一行能够进行移动或合并的操作。

对一行来说，对数据的操作分成两条线来操作，第一个条线是从第二个数据开始的，第二条线是从第一个数据开始的(k=0),并且只要a[i][k]>0,就会k++,即得二部分的对象变成下一个数值，这样可以保证第二条线的对象格子一直在第一条线的对象格子的左边。

一行行的循环主要依赖第一条线来进行的（for( ;j<4;j++)）,先判断除第一条线中对象格子中是否有数值。没有，则不进行任何操作。

如果第一条线中的对象格子有数值，同时第二条线中的对象没有数值，则将（第一条线中）该格子中的数与第二条线中的对象格子中的数值（第一个格子中的数）进行交换（arr[i][k]=arr[i][j];arr[i][j]=0;）同时k++,第二条线中的对象变成下一个数。

如果第一条线中的对象格子有数值，但是第二条线中对象格子中也有数值，则要进一步判断第二条线与第一条线中的对象数值是否相等，如果相等，则将第二条线中的数值乘二，第一条线中对象格子中的数值变成0，同时分数加上个子中等数值；如果第二条线与第一条线中的对象数值不相等，则将第二条线中的下一个对象（arr[i][++k]）与第一条线中的对象交换数值，同时将第一条线中的数值变成0。注意，此时的第二条线中的对象格子已经向右增加了一个（arr[i][++k]中最初的k 变成了 k+1）。

如此循环，第一条线中的对象也变成了下一个格子（for( ;j<4;j++)）,循环四次直到最后第一条线中的对象变为第四个格子。最后来一个大循环（for(int i=0;i<ROW;i++)）完成向左移的要求。

为了增强游戏体验，我组同学决定在移动过程中添加音效，但是一个函数只能返回一个数值，于是我们借用指针，定义Move函数时增加了一个形参（\* a）用于传递音效的信息

Game函数

Game函数是游戏over时进行后续游戏的函数，通过if语句先判断分数是否大于1万，如果小于1万，选择重新开始还是关闭游戏。大于一万进行邀请码输入或者扣除分数获得奖励

七、运行结果截图及解释

运行界面：

4x4 以及显示分数，通过system（）改变文字和界面颜色

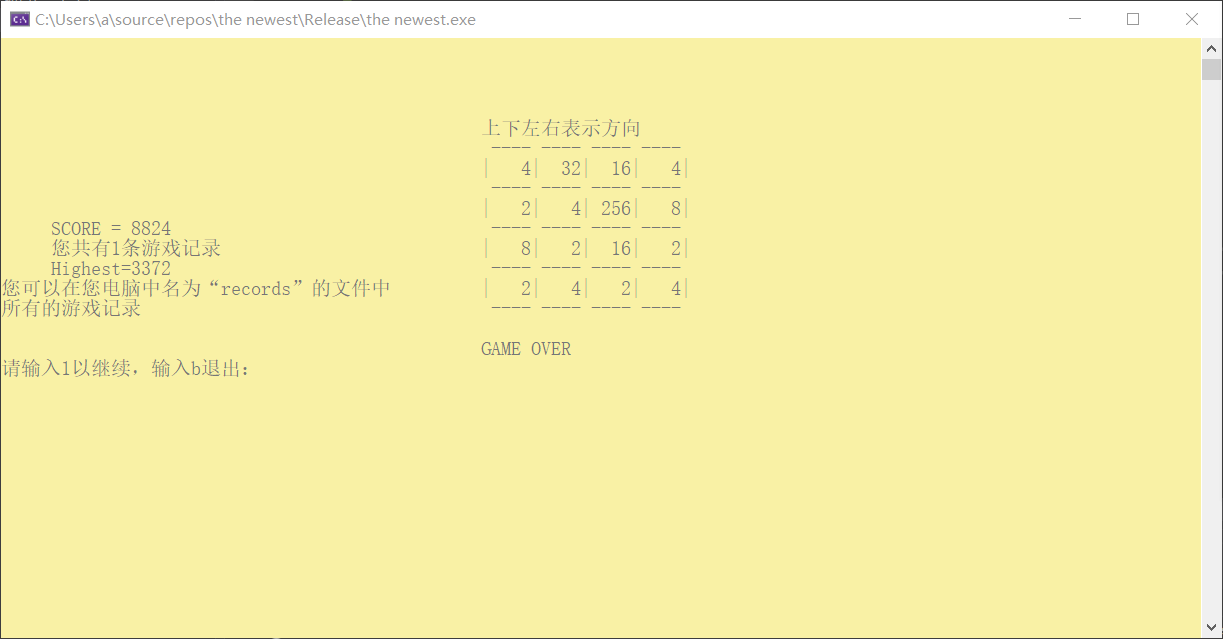
用 system("color 0A"); 其中color后面的0是背景色代号，A是前景色代号。各颜色代码如下：

0=黑色 1=蓝色 2=绿色 3=湖蓝色 4=红色 5=紫色 6=黄色 7=白色 8=灰色 9=淡蓝色 A=淡绿色 B=淡浅绿色 C=淡红色 D=淡紫色 E=淡黄色 F=亮白色

1、界面：4x4，显示分数和游戏记录、最高分。

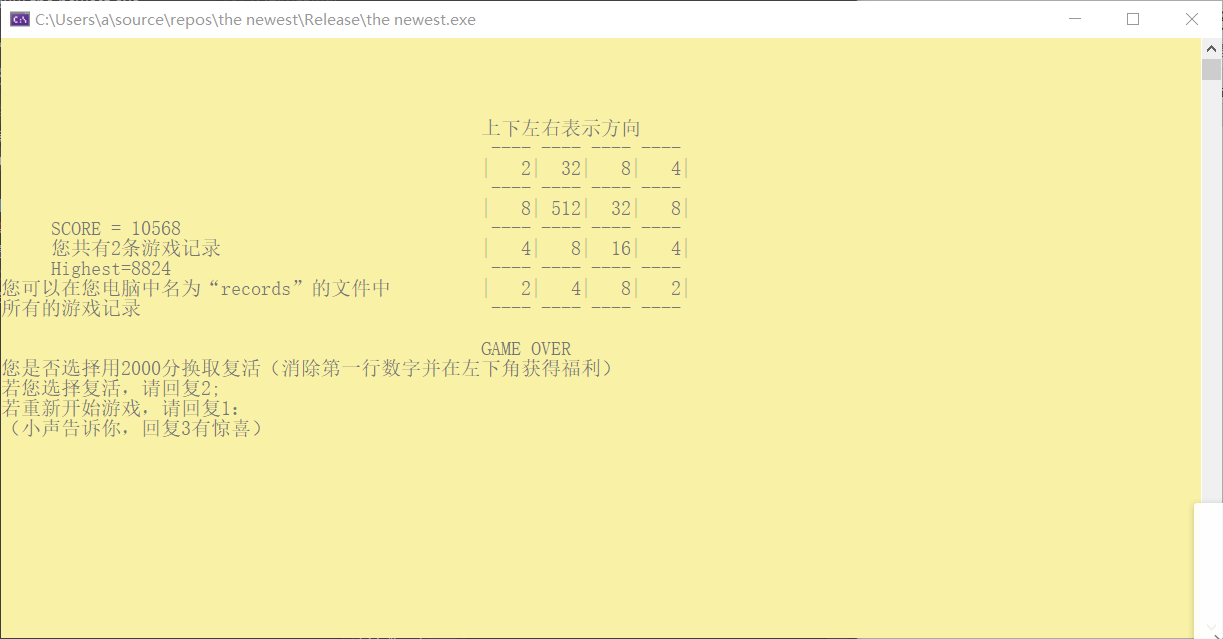


2、分数不超过10000，游戏结束后界面，只能选择是否清空重新开始游戏

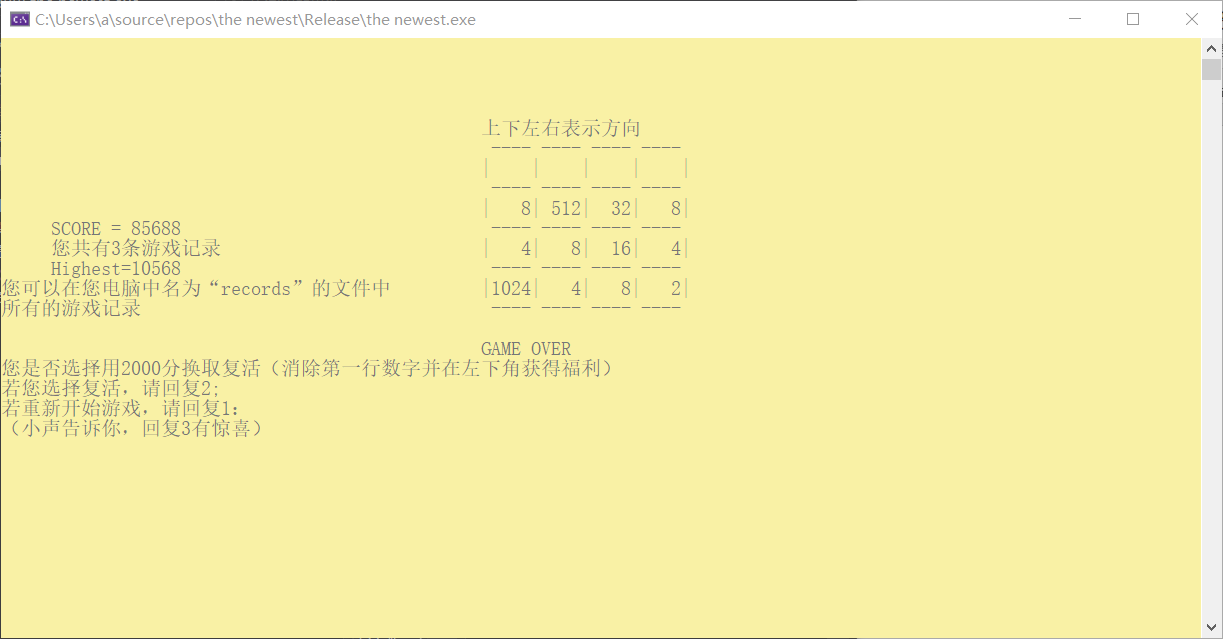


3、分数不小于10000分游戏结束画面：

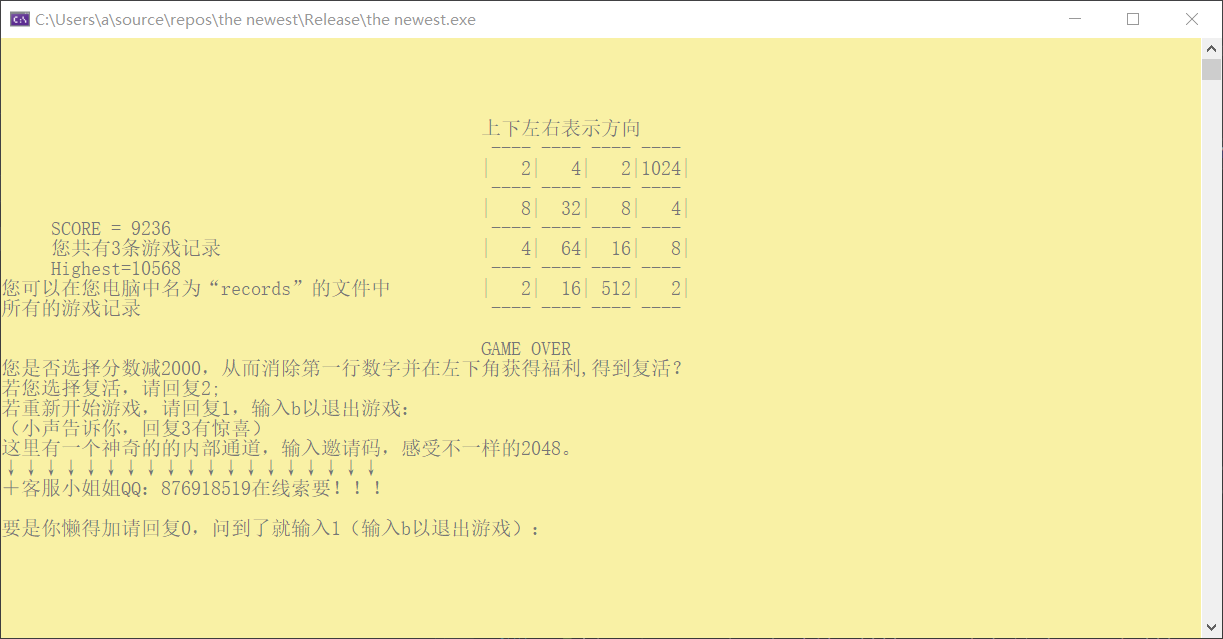
即有三个选择，消除1万分复活并获得彩蛋福利、重新开始游戏、选择是否输入邀请码。



4、输入2后，扣除一万分，获得福利后的界面：（第一行清零，左下变为1024）

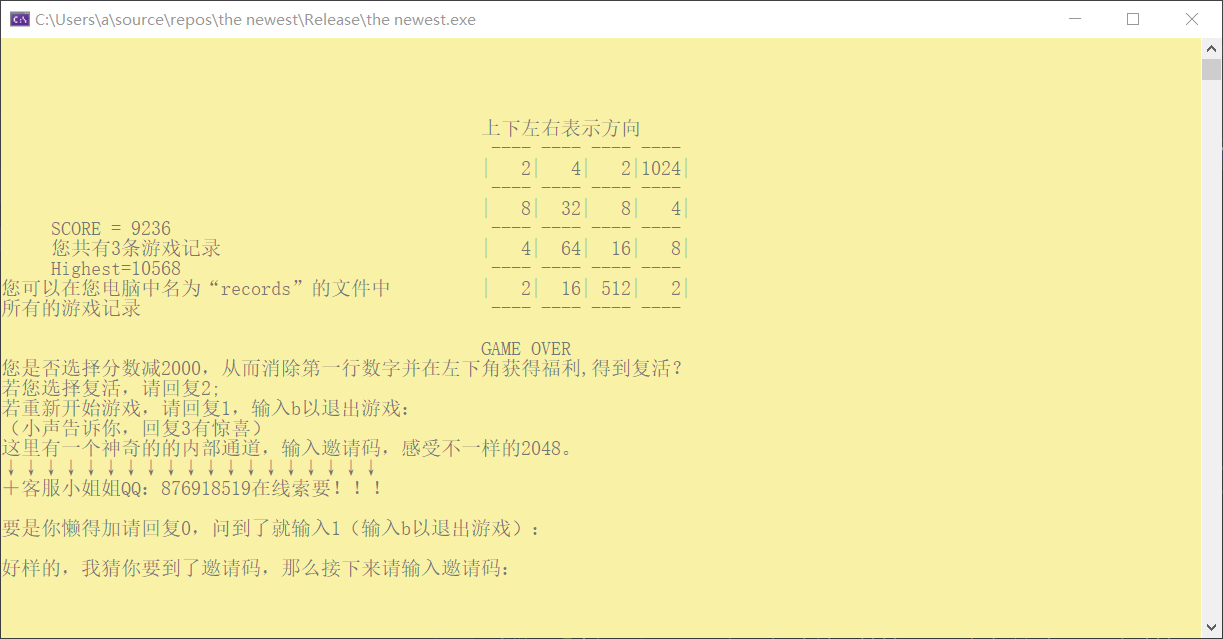


5、输入3后，选择是否进入邀请码这一渠道后的界面：

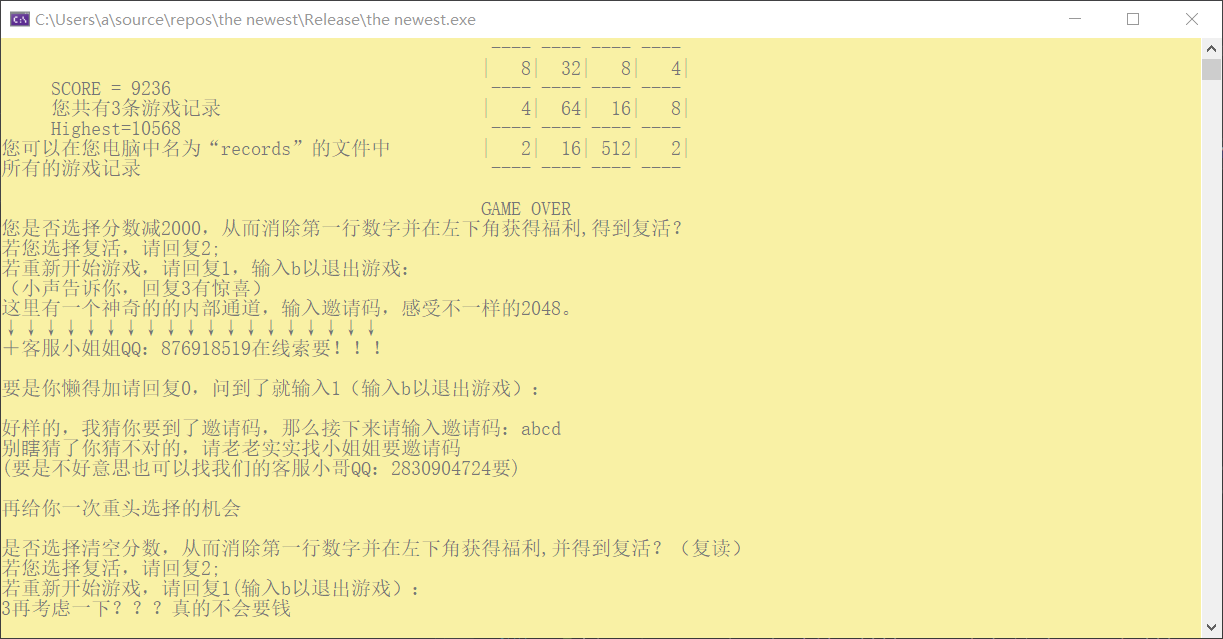


需要通过加客服qq876918519获取内部通道邀请码，输入邀请码后获得意想不到的快乐与快感。

6、输如1，选择输入邀请码后的界面：

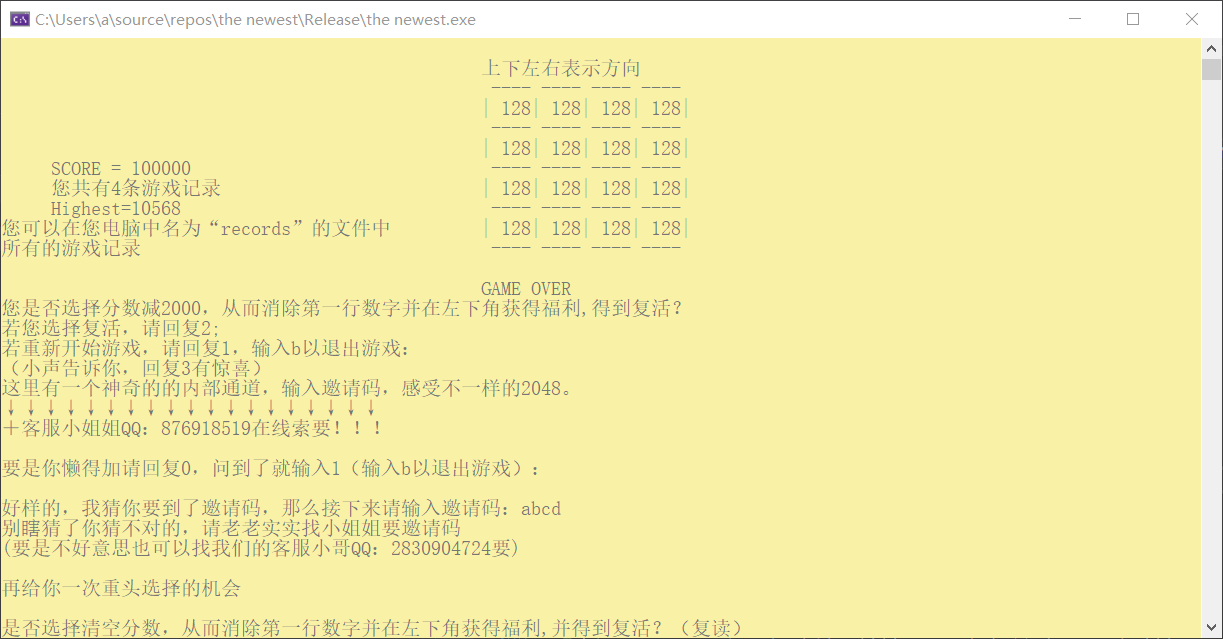


7、邀请码输如错误的界面：

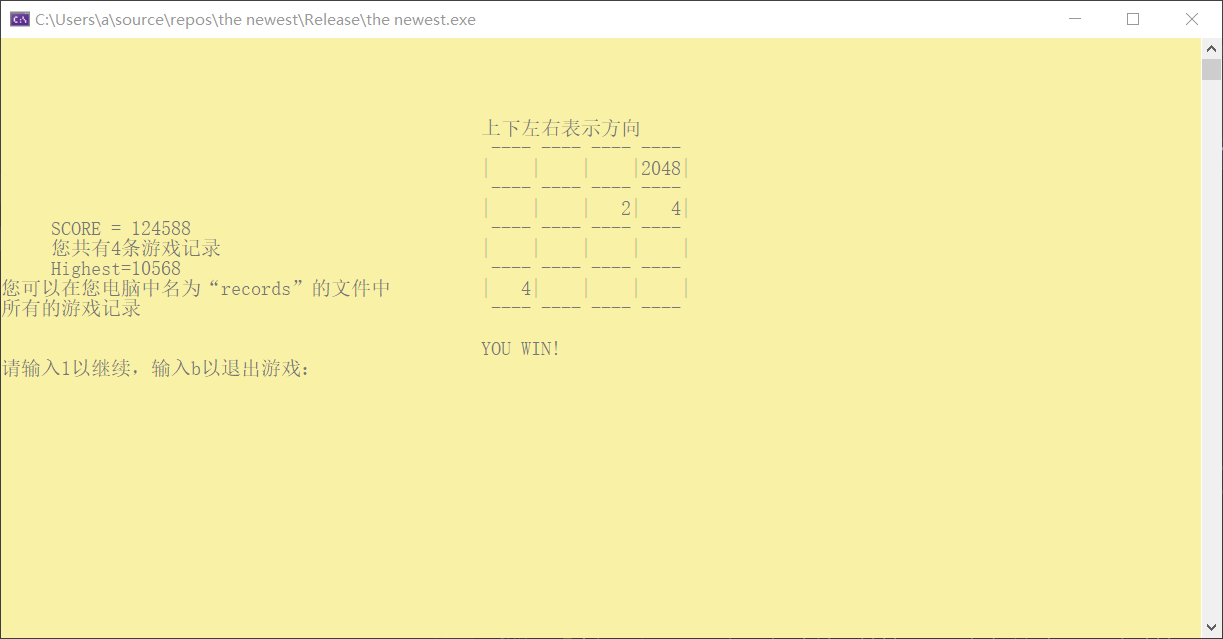


8、输如正确的邀请码：

界面全是128通过简单地上下移动十分轻松地获得2048数字方块，取得胜利



9、获取胜利界面：



**八、成员感想：**

曹成成：

在这次大作业中，我主要负责：前期程序框架构想以及搜集资料。部分思路有鉴于CSDN论坛。随后负责了优化程序中的：增加扣分获命的新玩法。以及流程图的绘制、编写活动总结。

在这次大作业中，我最大的感受就是：团队的集体力量远比一个人苦苦思索大得多。在这次大作业中，我们进行了线上线下的多次讨论。约研修室很多次以及线上语音交流几小时很多次。大家集中到一起，一起钻研程序，查阅资料，想点子。思考如何优化最初的这个小游戏。想了很多办法，也修改了10几次。最终结果我们还是很满意的。

一个函数一个函数的细扣，以及各小组成员分工写函数，最后在主函数中调用。让我再一次体会到了模块化这一概念。这也是c语言的灵活之处。我想通过这次大作业。我爱上了编程。

回味着好几个从7点到10.30在图书馆写程序的日子，是蛮充实的。

这次程序 用到了许多**所学知识**

**多重循环嵌套（打印界面）、如何隐藏光标、二维数组的运用、指针的巧妙运用（主要是队友设计）、流与文件(自学+运用到程序记录游戏次数和最高分中)、定义全局变量、调用多个函数、字符的输入输出、结构体的运用（隐藏光标时）**

所以这个程序展现了我们活学活用哈哈。

但是这个程序还是可以更加提升的，比如界面优化，增加图形，但这是在c++环境下比较容易实现。以及未能添加链表到程序中。

郑洋：

尽管这次实验作业中遇到了许多困难，比如光标的定位问题，但我们通过在网上搜索资料和亲身实践最终将其调试成功了，这对我们来说无疑是一种巨大的成就，毕竟这些内容在课上、在书本上都没有涉及，是靠别人的经验和我们自己的努力实现的，对此我感到很自豪。

夏诗怡：

这次写程序应该是让我最有成就感的一次了，每一行代码都在讨论和斟酌后亲手敲下，不断改进，不断学习，用已学会的和即将学会的知识慢慢把想法都变成程序中的现实。在学习新知识的过程中我深刻体会到，实践才是检验真理的唯一标准，不同的编译器里同样的函数使用方法也有细微差别，在不断尝试不断探索中学会使用，也许才是我们学习的最佳方法。小组合作也十分愉快，在线上线下无数次思维碰撞中我们把最初的粗糙代码一改再改，代码也从三多行改到五百多行，每个人都在努力把自己负责的部分做到极致，出现错误也会奥秘全局考虑来调整，写代码、改代码很累，但看到小组成员的认真尽力，看到运行结果的进步，我们的付出值得且令人快乐。

李筱婷：

在设计程序时首先要有一个明晰的思路，再一步步填充，填充过程中不断发现问题、解决问题，敢于提出不足和漏洞，以及可以添加的创新创意。只要敢想，就可以通过从各种渠道的学习去实现它；只要敢尝试，就可以从错误中不断修正从而达到目的。在完成大作业的过程中，我不仅体会到了将所学落实到真正意义的程序中去的成就感，我还感受到了小组内部合理分工、团结协作的快乐。小组成员将各自的任务落实好之后还会帮助其他成员检查问题并提出解决方案，通过线上qq交流、线下相约研修室，共同商讨、共同努力、共同进步。

刘一经：

经过我们组全队员的多次共同讨论，经过不断的debug和努力，最终的作业效果基本满足最初的设定。团队成员的明确分工使我我感受到了团队合作的力量和团队合作的强大所在!