同济大学

实验报告

彩球游戏（消消乐）

装

订

线

计科-3班

1553449 王志业

完成日期：2017.3.30

装 订 线

1. 彩球游戏（消消乐）基本介绍及要求

1.1. 游戏规则

游戏区域大小自选（5*5-9*9），初始随机生成9种不同颜色的球占满全部空间。游戏开始后，可交换的小球会以不同的形状显示出来，根据提示鼠标单击选中可以交换两个相邻小球的位置，若交换后任意行（列或行列同时满足）满3个及以上可消除，得分为消除个数（初始生成后的消除不得分）。消除后上方的彩球垂直下落，上方的空余位置随机补充彩球至满，补充后满足要求则自动消除并积分。无任何可移动的小球则结束游戏。

1.2. 游戏显示要求

可移动位置的彩球、被选中的彩球要有不同的显示效果；彩球下落时要有彩球沿着通路进行移动的动画效果；消除时应有相应的动画效果。

2. 整体设计思路

将该游戏分为两大部分即：内部数组实现与cmd图形界面及动画效果的的构建。

2.1. 内部数组实现

自定义类型ball，里面包括三个整形数据cor（颜色）、dis（是否可消除）、cha（是否可交换）。在用户通过函数int_xy输入行（row）与列（line）后，建立row*line的ball类型矩阵。根据系统时间生成伪随机数种子，通过函数set_ball将1-9的随机数赋值给cor。然后寻找可消除的小球（**find_dis**函数）->内部ball数组cor置零（disappear函数）->上面的非零cor数据向下填补（**up_down**）->在cor为零位置填补新随机数据(reload)->(前面步骤反复循环再无可直接消除的小球后)->寻找可通过和临近小球交换即可消除的小球，将其cha置为1（**find_cha**）->通过鼠标交换两球内部数组的cor值->将整个数组的cha,dis值重新置零后返回第一步循环直至再无可交换的小球。

2.2. Cmd图形界面构建及动画效果实现

Cmd图形界面的构建主要依靠cmd_console_tools里的函数，通过规划每个字符“┌┬┐├┼┤└┴┘”形状出现的位置，来形成完整的带分割线的界面，填入小球“○”，以对应位置上的cor值作为颜色。

动画效果的建立，消除动画：可通过在该消除位置上显示“□”，静止100ms在消失实现消除的动画效果。移动动画:将球不断的在运动路径上的分割线上显示出来，直到达到预定的位置。

鼠标的控制：通过计算屏幕上行列与二维数组里位置的对应关系，取鼠标的位置显示在屏幕下方（如当前光标位置H行5列），鼠标左键单击为选择某小球，再次选择临近小球则交换位置，选择无关小球无效。

3. 主要功能的实现

3.1重要函数解析

3.1.1 find_dis函数

本函数功能为针对二维数组的某一特定元素，寻找其所在行列是否满足消除要求，若满足，则该位置的dis值置为1，返回值为消除小球的个数。需要参数：ball类型二维数组，目标行列值tar_x,tar_y，二位数组的行列总数row,line。函数内部定义行列的可消除数xcount,ycount及计数变量i。

行循环，先往右找，若右边cor值相等，则xcount++，dis++，若不等则退出循环。往左边同理。

行循环完成后，若xcount值大于3，则证明行内满足消除要求。若小于3，则xcount置零Dis>0的再dis--。列循环同理。

示例：

以此数组为例，当执行find_dis(inside_array,2,6,8,8)时，即C7位置，第一个循环向右判断不相等直接退出。进入第二个循环，向左判断相等，i--，xcount++，inside_array[2][5].dis++，后再判断左边第二个，不相等，退出循环。判断xcount是否小于3，本次小于3，则xcount置零。将刚才自加过的dis再自减。此时判断列，上下各有一个7，两个循环结束后ycount=3，大于3，返回xcount+ycount 的值。

```
//寻找数组中某一位置相邻行列中三个及以上颜色相同的,并统计可消除个数,返回消除球的个数。
int find_dis(ball inside_array[][9], int tar_x, int tar_y, int row, int line)
{
    int xcount = 0, ycount = 0;
    int i;
    xcount = 1;
    ycount = 1;
    inside_array[tar_x][tar_y].dis++;
    for (i = 1; i < row - tar_x; i++)
    {
        if (inside_array[i + tar_x][tar_y].cor == inside_array[tar_x][tar_y].cor)
        {
            xcount++;
            inside_array[i + tar_x][tar_y].dis++;
        }
        else
            break;
    }
    for (i = -1; i >= -tar_x; i--)
    {
        if (inside_array[i + tar_x][tar_y].cor == inside_array[tar_x][tar_y].cor)
        {
            xcount++;
            inside_array[i + tar_x][tar_y].dis++;
        }
        else
            break;
```

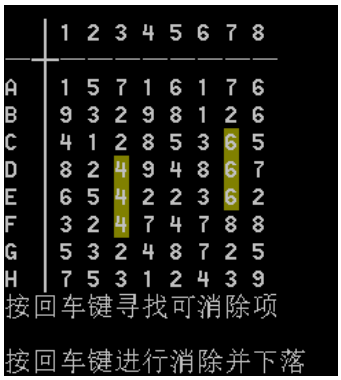
3.1.2 up_down函数

本函数属于递归函数，功能为将cor为零的上面的位置对应的非零cor值挪到下方零位置。若该位置也为零，则再次进入up_down函数。直到tar_x参数小于0，即找到最顶端时。需要的参数为ball类型的二维数组，待执行该函数的位置tar_x,tar_y（一般需外部判断cor是否为零，为

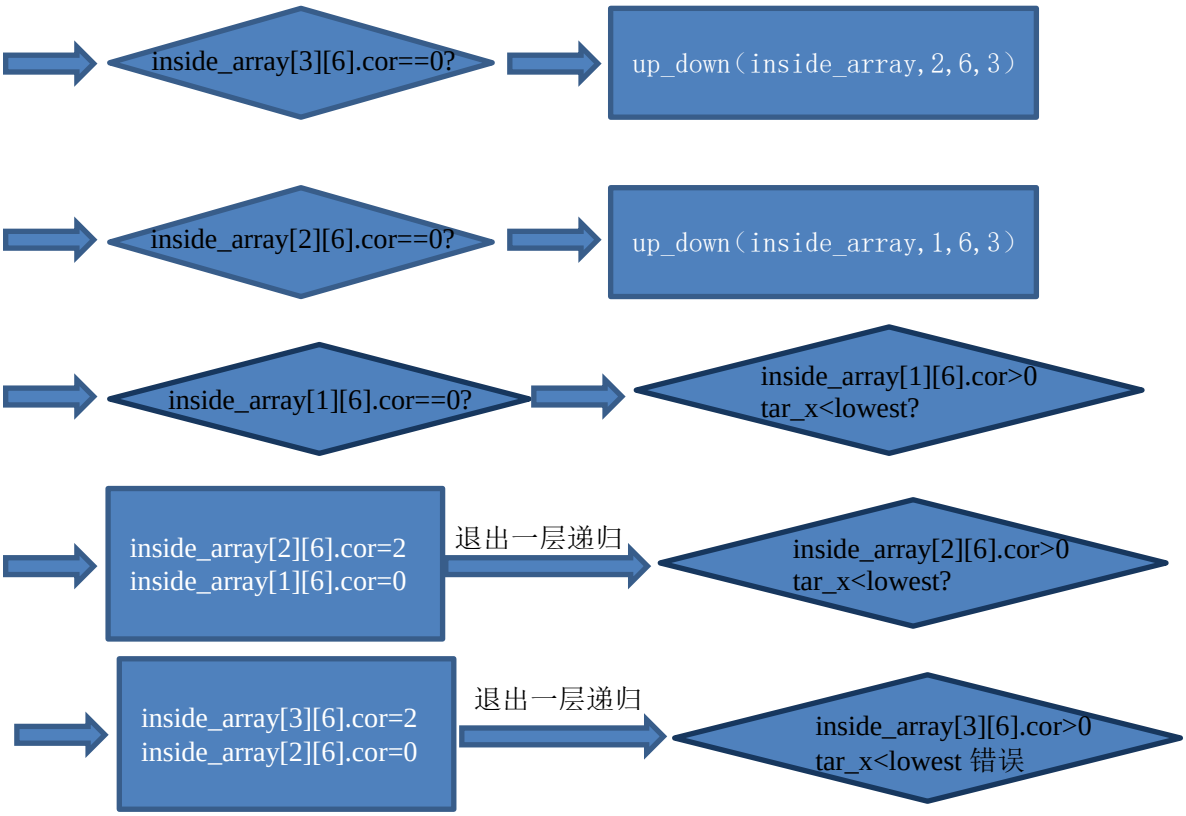
零则进入该函数），以及最低行位置，即循环到结尾不能把值赋到tar_x+1行。

```
void up_down1(ball inside_array[][9], int tar_x, int tar_y, int lowest)
{
    if (tar_x < 0)
        return;
    if (inside_array[tar_x][tar_y].cor == 0)
    {
        up_down1(inside_array, tar_x - 1, tar_y, lowest);
    }
    if (inside_array[tar_x][tar_y].cor > 0 && tar_x < lowest)
    {
        inside_array[tar_x + 1][tar_y].cor = inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor = 0;
    }
}
```

代码运行流程：



以上图数组为例，标记黄色处为可消除项将被置零。此时若针对D7处执行函数up_down（up_down（inside_array,3,6,3））,流程如下：



3.1.3 find_cha函数

功能:判断目标位置小球是否能通过上下左右互换达到可消除的状态。需要参数ball类型二维数组，目标行列值tar_x,tar_y,数组行列值row,line。过程为将目标位置cor值分别与上下左右的cor值交换，对交换后位置find_dis判断是否达到可消除的状态，若达到，则将目标位置的cha值自加。

示例：

以B6做目标位置：find_cha(inside_array,1，5，8，8)

函数中四个if判断是为避免出现在边角时没有上下左右位置导致错误。首先交换到A6位置，做find_dis(inside_array,1，5，8，8)判断，如果inside_array[1][5].dis>0,则证明可以交换，显然在这里不行。再判断…直到与B7交换时，做find_dis(inside_array,1，5，8，8)判断，且inside_array[1][5].dis>0，则B6、B7符合交换要求，将两位置上的cha置为1。

3.2菜单第八项功能流程解析

游戏开始后，调用draw3画出带分割线的图形界面。进入初始循环：消除掉初始可消除的小球,下落,再次消除直到所有位置inside_array[i][j].dis=0,退出循环。此时打开积分器（score）和鼠标控制（input_mouse）,进入下一个循环

寻找可消除的小球（find_dis函数）->

内部ball数组cor置零（disappear函数）演示消除动画（dis_act函数）->

上面的非零cor数据向下填补（up_down）->

在cor为零位置填补新随机数据(reload)->

(前面步骤反复循环再无可直接消除的小球后)->

寻找可通过和临近小球交换即可消除的小球，将其cha置为1（find_cha）->

通过鼠标交换两球内部数组的cor值->

将整个数组的cha,dis值重新置零->

（以上步骤无限次循环直至再无可交换的小球）

4. 调试过程碰到的问题

4.1 up_down函数

在调试up_down函数时，由于递归，到最外面一层递归时会将目标位置下面的cor值更改。

解决问题时引入了一个新的参数lowest，代表起始的行数，如果超过这个值后就不再进行将当前位置值赋值到下一行的操作，从而顺利解决了问题。

4.2 setcolor函数、gotoxy函数

在使用完其他的颜色后，必须要立即把颜色改回BLACK和WHITE，否则在下次输出时颜色会混乱。调试时经常会出现在正文“按任意键继续”，即用array_out输出数组在回去为其更改颜色后，需将光标gotoxy到无关的地方再进行输出。

5. 心得体会

这是我转到电信学院的第一次大作业，在边学边做中提高了很多:经历了某个项目从开始设计到实现到调试的过程，了解如何把一个大项目拆分成一个个好实现的小项目，积累了很多编程经验。同时我也发现了我的不足之处，代码不够“漂亮”，重复的地方，混杂的地方太多，函数设计的不明确。其实我是到了最后才明白过来应该先把函数设计好功能明确之后再开始编码，也算一个经验吧。

另外还要特别鸣谢曾果同学，在我尝试写上学期第二次大作业毫无头绪时把他的代码拿出来给我阅读对我指导，有很多地方看了都有恍然大悟的感觉，这次的大作业里也有一部分借鉴了他的方法。

现在觉得编程是件很有乐趣的事，通过算法，代码去解决实际问题，那种解决问题后的快感是别的专业难以带给我的。希望以后自己能在这个专业潜心学习，stay hungry,stay foolish.能坚持自己的兴趣，坚持每日thinking and coding.（先给自己定个小目标，达到能被老师鄙视的水平哈哈）

6. 附件：源程序

```
#pragma once
#include<iomanip>
#include <conio.h>
#include <windows.h>
#include <stdlib.h>
#include <time.h>
#include <iostream>
#include "cmd_console_tools.h"
#include <fstream>

using namespace std;
struct ball
{

    int cor;//颜色
    int dis;//是否可消除
    int cha;//是否可互换
};
void int_xy(int *row, int *line);

//初始生成，内部数组
void set_ball(ball inside_array[][9], int row, int line);

//寻找数组中某一位置相邻行列中三个及以上颜色相同的,并统计可消除个数,返回消除球的个数。
int find_dis(ball inside_array[][9], int tar_x, int tar_y, int row, int line);

//输出内部数组至屏幕
void array_out(ball inside_array[][9], int row, int line);

void draw0(ball inside_array[][9], int row, int line);

//指定位置若连成三个及以上将内部数组对应位置置0
void disappear(ball inside_array[][9], int row, int line);

//碰到位置为0则将上面的数组值（不为0）赋值到下面，若上面为0则再往上找,并挪下出现动画
void up_down(ball inside_array[][9], int tar_x, int tar_y, int lowest);
```

```
//同上，但无动画
void up_down1(ball inside_array[][9], int
tar_x, int tar_y, int lowest);

//从最后一行往前数，若有位置为0，则停下进行
up_down判断
void array_iszero(ball inside_array[][9],
int row, int line);

//在指定位置重新生成小球
void reload(ball inside_array[][9], int
tar_x, int tar_y, int row, int line);

//将一个数组的dis值重置为0
void reset_dis(ball inside_array[][9], int
row, int line);

//将一个数组的cha值重置为0
void reset_cha(ball inside_array[][9], int
row, int line);

//找出某一位置所有可以互换的彩球
void find_cha(ball inside_array[][9], int
tar_x, int tar_y, int row, int line);

//画出界面，无分割线
void draw2(ball inside_array[][9], int row,
int line);

//画出界面有分割线
void draw3(ball inside_array[][9], int row,
int line);

//可消除小球标明
void draw4(ball inside_array[][9], int row,
int line);

//消除并附加动画
void dis_act(ball inside_array[][9], int row,
int line);

//落下并填补附加动画
void down_reload(ball inside_array[][9],
int row, int line);

//标出可移动的球,若不可移动则变回普通形状
void mark_ball(ball inside_array[][9], int
row, int line);

//交换两球的位置并出现动画
void ball_change(ball inside_array[][9],
int x, int y, int ex, int ey);

//读取鼠标位置及鼠标指令
int mouse_input(const HANDLE hout, ball
inside_array[][9], int*pY, int*pX, int*pEY,
int*pEX, int row, int line);


//记录消除的分数（初始消除不得分）
void count_score(const HANDLE hout, ball
inside_array[][9], int row, int line, int
*pscore);

void menu();

void q1(HANDLE hout, ball inside_array[][9],
int *row, int *line);

void q2(HANDLE hout, ball inside_array[][9],
int *row, int *line);

void q3(HANDLE hout, ball inside_array[][9],
int *row, int *line);

void q4(HANDLE hout, ball inside_array[][9],
int *row, int *line);
void q5(HANDLE hout, ball inside_array[][9],
int *row, int *line);

void q6(HANDLE hout, ball inside_array[][9],
int *row, int *line);

void q7(HANDLE hout, ball inside_array[][9],
int *row, int *line);

void q8(HANDLE hout, ball inside_array[][9],
int *row, int *line);

void q9(HANDLE hout, ball
inside_array[][9]);
//寻找数组中某一位置相邻行列中三个及以上颜
色相同的,并统计可消除个数,返回消除球的个数。
int find_dis(ball inside_array[][9], int
tar_x, int tar_y, int row, int line)
{
    int xcount = 0, ycount = 0;
    int i;
    xcount = 1;
    ycount = 1;
    inside_array[tar_x][tar_y].dis++;
    for (i = 1; i < row - tar_x; i++)
    {
        if (inside_array[i +
tar_x][tar_y].cor ==
inside_array[tar_x][tar_y].cor)
        {
            xcount++;
```

```
            inside_array[i +
tar_x][tar_y].dis++;
        }
        else
            break;
    }
    for (i = -1; i >= -tar_x; i--)
    {
        if (inside_array[i +
tar_x][tar_y].cor ==
inside_array[tar_x][tar_y].cor)
        {
            xcount++;
            inside_array[i +
tar_x][tar_y].dis++;
        }
        else
            break;
    }
    if (xcount < 3)
    {
        xcount = 0;
        inside_array[tar_x][tar_y].dis--;
        if (inside_array[tar_x +
1][tar_y].dis>0 && inside_array[tar_x +
1][tar_y].cor ==
inside_array[tar_x][tar_y].cor)
            inside_array[tar_x +
1][tar_y].dis--;
        if (inside_array[tar_x -
1][tar_y].dis > 0 && inside_array[tar_x -
1][tar_y].cor ==
inside_array[tar_x][tar_y].cor)
            inside_array[tar_x -
1][tar_y].dis--;
    }
    if (xcount == 0)
        inside_array[tar_x][tar_y].dis++;
    for (i = 1; i < line - tar_y; i++)
    {
        if (inside_array[tar_x][i +
tar_y].cor ==
inside_array[tar_x][tar_y].cor)
        {
            ycount++;
            inside_array[tar_x][i +
tar_y].dis++;
        }
        else
            break;
    }
    for (i = -1; i >= -tar_y; i--)
    {
        if (inside_array[tar_x][i +
tar_y].cor ==
inside_array[tar_x][tar_y].cor)
        {
            ycount++;
            inside_array[tar_x][i +
tar_y].dis++;
        }
        else
            break;
    }
    if (ycount < 3)
    {
        ycount = 0;
        if (xcount == 0)

    inside_array[tar_x][tar_y].dis--;
        if (inside_array[tar_x][tar_y -
1].dis>0 && inside_array[tar_x][tar_y -
1].cor == inside_array[tar_x][tar_y].cor)
            inside_array[tar_x][tar_y -
1].dis--;
        if (inside_array[tar_x][tar_y +
1].dis > 0 && inside_array[tar_x][tar_y +
1].cor == inside_array[tar_x][tar_y].cor)
            inside_array[tar_x][tar_y +
1].dis--;
    }
    /*} */
    if (ycount >= 3 && xcount >= 3)
        return xcount + ycount - 1;
    return xcount + ycount;

}
//指定位置若连成三个及以上将内部数组对应位
置置0
void disappear(ball inside_array[][9], int
row, int line)
{
    int i, j;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < line; j++)
        {
            if (inside_array[i][j].dis>0)
                inside_array[i][j].cor =
0;
        }
    }
}
//碰到位置为0则将上面的数组值（不为0）赋值
到下面，若上面为0则再往上找,并挪下出现动画
void up_down(ball inside_array[][9], int
tar_x, int tar_y, int lowest)
{
```

```
    const HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
    if (tar_x < 0)
        return;
    if (inside_array[tar_x][tar_y].cor ==
0)
    {
        up_down(inside_array, tar_x - 1,
tar_y, lowest);
    }
    if (inside_array[tar_x][tar_y].cor >0
&& tar_x < lowest)
    {
        inside_array[tar_x + 1][tar_y].cor
= inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
0;
        showstr(hout, 2 + tar_y * 4, (tar_x
+ 1) * 2, "  ", COLOR_HWHITE, COLOR_HWHITE);

        Sleep(50);//延时0.05秒
        showstr(hout, 2 + tar_y * 4, (tar_x
+ 1) * 2 + 1, "○", inside_array[tar_x +
1][tar_y].cor, COLOR_HWHITE);
        Sleep(50);
        showstr(hout, 4 * tar_y + 2, tar_x
* 2 + 3, "—", COLOR_HWHITE, COLOR_BLACK);
        Sleep(50);
        showstr(hout, 2 + tar_y * 4, (tar_x
+ 2) * 2, "○", inside_array[tar_x +
1][tar_y].cor, COLOR_HWHITE);
    }

}
//同上，但无动画
void up_down1(ball inside_array[][9], int
tar_x, int tar_y, int lowest)
{
    if (tar_x < 0)
        return;
    if (inside_array[tar_x][tar_y].cor ==
0)
    {
        up_down1(inside_array, tar_x - 1,
tar_y, lowest);
    }
    if (inside_array[tar_x][tar_y].cor > 0
&& tar_x < lowest)
    {
        inside_array[tar_x + 1][tar_y].cor
= inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
0;
    }
}
//在特定位置填入一个随机颜色的球
void reload(ball inside_array[][9], int
tar_x, int tar_y, int row, int line)
{
    srand((unsigned int)time(0));
    if (inside_array[tar_x][tar_y].cor ==
0)
        inside_array[tar_x][tar_y].cor =
rand() % 9 + 1;
}
//将一个数组的dis值重置为0
void reset_dis(ball inside_array[][9], int
row, int line)
{
    int i, j;
    for (i = 0; i < row; i++)
        for (j = 0; j < line; j++)
            inside_array[i][j].dis = 0;
}
//将一个数组的cha值重置为0
void reset_cha(ball inside_array[][9], int
row, int line)
{
    int i, j;
    for (i = 0; i < row; i++)
        for (j = 0; j < line; j++)
            inside_array[i][j].cha = 0;
}
//找出某一位置所有可以互换的彩球
void find_cha(ball inside_array[][9], int
tar_x, int tar_y, int row, int line)
{
    int temp;
    if (tar_x > 0)
    {
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
inside_array[tar_x - 1][tar_y].cor;
        inside_array[tar_x - 1][tar_y].cor
= temp;
        if (find_dis(inside_array, tar_x,
tar_y, row, line)>0)
        {

    inside_array[tar_x][tar_y].cha = 1;
            inside_array[tar_x -
1][tar_y].cha = 1;
        }
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
```

装 订 线

```
inside_array[tar_x - 1][tar_y].cor;
        inside_array[tar_x - 1][tar_y].cor
= temp;
    }
    reset_dis(inside_array, row, line);
    if (tar_x < row - 1)
    {
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
inside_array[tar_x + 1][tar_y].cor;
        inside_array[tar_x + 1][tar_y].cor
= temp;
        if (find_dis(inside_array, tar_x,
tar_y, row, line)>0)
        {

    inside_array[tar_x][tar_y].cha = 1;
            inside_array[tar_x +
1][tar_y].cha = 1;
        }
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
inside_array[tar_x + 1][tar_y].cor;
        inside_array[tar_x + 1][tar_y].cor
= temp;
    }
    reset_dis(inside_array, row, line);
    if (tar_y > 0)
    {
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
inside_array[tar_x][tar_y - 1].cor;
        inside_array[tar_x][tar_y - 1].cor
= temp;
        if (find_dis(inside_array, tar_x,
tar_y, row, line) > 0)
        {

    inside_array[tar_x][tar_y].cha = 1;
            inside_array[tar_x][tar_y -
1].cha = 1;
        }
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
inside_array[tar_x][tar_y - 1].cor;
        inside_array[tar_x][tar_y - 1].cor
= temp;
    }
    reset_dis(inside_array, row, line);
    if (tar_y < line - 1)
    {
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
inside_array[tar_x][tar_y + 1].cor;
        inside_array[tar_x][tar_y + 1].cor
= temp;
        if (find_dis(inside_array, tar_x,
tar_y, row, line)>0)
        {

    inside_array[tar_x][tar_y].cha = 1;
            inside_array[tar_x][tar_y +
1].cha = 1;
        }
        temp =
inside_array[tar_x][tar_y].cor;
        inside_array[tar_x][tar_y].cor =
inside_array[tar_x][tar_y + 1].cor;
        inside_array[tar_x][tar_y + 1].cor
= temp;
    }
    reset_dis(inside_array, row, line);
}
//输入行和列的值（公用）
void int_xy(int *row, int *line)
{
    while (1)
    {
        cout << "please input the number of
row(5-9)" << endl;
        cin >> *row;
        if (cin.fail())
        {
            cout << "you have input a wrong
number...";
            cin.clear();
            cin.ignore(1000, '\n');
            continue;
        }

        if (*row >= 5 && *row <= 9)
        {
            cin.ignore(1000, '\n');
            break;
        }
        cout << "you have input a wrong
number...";
        cin.clear();
        cin.ignore(1000, '\n');

    }
    while (1)
    {
```

```
        cout << "please input the number of row(5-9)" << endl;
        cin >> *line;
        if (cin.fail())
        {
            cout << "you have input a wrong number...";
            cin.clear();
            cin.ignore(1000, '\n');
            continue;
        }

        if (*line >= 5 && *line <= 9)
        {
            cin.ignore(1000, '\n');
            break;
        }
        cout << "you have input a wrong number...";
        cin.clear();
        cin.ignore(1000, '\n');

    }
}
//初始生成，内部数组
void set_ball(ball inside_array[][9], int row, int line)
{
    srand((unsigned int)time(0));
    int i, j;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < line; j++)
        {
            inside_array[i][j].cor = rand() % 9 + 1;
            inside_array[i][j].dis = 0;

        }
    }
}
//输出内部数组
void array_out(ball inside_array[][9], int row, int line)
{
    int i, j;
    cout << endl << "  " << "|";
    for (i = 1; i <= line; i++)
        cout << i << " ";
    cout << endl;
    cout << "—" << "┼";
    for (i = 1; i <= line; i++)
        cout << "—";
    cout << endl;
    for (i = 0; i < row; i++)
    {
        cout << char(i + 'A') << " " << "|";
        for (j = 0; j < line; j++)
            cout << inside_array[i][j].cor << " ";
        cout << endl;
    }
}
//读取鼠标位置及鼠标指令
int mouse_input(const HANDLE hout, ball inside_array[][9], int*pY, int*pX, int*pEY, int*pEX, int row, int line)
{
    const HANDLE hin = GetStdHandle(STD_INPUT_HANDLE);  //取标准输入设备对应的句柄

    int X = 0, Y = 0, mark = 0;//x stand for the line and y stand for the row;
    int action;
    enable_mouse(hin);

    while (1)
    {
        action = read_mouse(hin, X, Y);
        if (Y % 2 == 0 && Y / 2 >= 1 && Y / 2 <= row && (X % 4 == 2 || X % 4 == 3) && X / 4 >= 0 && X / 4 < line)
        {
            gotoxy(hout, 0, row * 2 + 2);
            cout << "[当前光标] " << setw(2) << char(Y / 2 - 1 + 'A') << "行" << setw(2) << char(X / 4 + '1') << "列        ";


            if (action == MOUSE_LEFT_BUTTON_CLICK&&mark == 0)
            {

                *pY = Y / 2 - 1;
                *pX = X / 4;
                showstr(hout, 2 + (X / 4) * 4, Y, "⊙", inside_array[*pY][*pX].cor, COLOR_HWHITE);
                mark = 1;
                setcolor(hout, COLOR_BLACK, COLOR_WHITE);
                continue;
            }

            if (action == MOUSE_LEFT_BUTTON_CLICK&&mark == 1)
            {
```

```
                    *pEY = Y / 2 - 1;
                    *pEX = X / 4;
                    if ((*pX == *pEX || *pY == *pEY) && (*pX + *pY - *pEX - *pEY == 1 || *pX + *pY - *pEX - *pEY == -1))
                    {

    ball_change(inside_array, *pX, *pY, *pEX, *pEY);
                        gotoxy(hout, 0, row * 2 + 2);
                        setcolor(hout, COLOR_BLACK, COLOR_WHITE);
                        break;
                    }
                    else
                    {
                        mark = 0;
                        showstr(hout, 2 + *pX * 4, 2 * (1 + *pY), "○", inside_array[*pY][*pX].cor, COLOR_HWHITE);
                    }

                }

                if (action == MOUSE_RIGHT_BUTTON_CLICK)
                    return 1;
            }

        }

        return 0;
}
//记录消除的分数（初始消除不得分）
void count_score(const HANDLE hout, ball inside_array[][9], int row, int line, int *pscore)
{
    int count = 0, i, j;
    for (i = 0; i < row; i++)
        for (j = 0; j < line; j++)
        {
            if (inside_array[i][j].cor == 0)
                count++;
        }
    *pscore += count;
    gotoxy(hout, 16, 0);
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    cout << "your score:" << *pscore << setw(4) << "按鼠标右键退出";
}
```

```
//draw the color balls without fenge line
void draw2(ball inside_array[][9], int row, int line)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    system("cls");
    setconsoleborder(hout, 35, row + 6, row + 6);
    setfontsize(hout, L"新宋体", 20);
    setcursor(hout, CURSOR_INVISIBLE);
    cout << "屏幕：" << row + 6 << "行" << 35 << "列" << endl;


    showstr(hout, 0, 1, "┌", COLOR_HWHITE, COLOR_BLACK);
    for (int i = 0; i < line; i++)
        showstr(hout, i * 2 + 2, 1, "—", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 2 * line + 2, 1, "┐", COLOR_HWHITE, COLOR_BLACK);

    for (int j = 0; j < row; j++)
    {
        showstr(hout, 0, 2 + j, "|", COLOR_HWHITE, COLOR_BLACK);
        showstr(hout, 2 * (line + 1), 2 + j, "|", COLOR_HWHITE, COLOR_BLACK);
    }


    showstr(hout, 0, row + 2, "└", COLOR_HWHITE, COLOR_BLACK);
    for (int i = 0; i < line; i++)
        showstr(hout, 2*i+2, row+2, "—", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 2 * (line + 1), row + 2, "┘", COLOR_HWHITE, COLOR_BLACK);




    for (int i = 0; i < row; i++)
        for (int j = 0; j < line; j++)
        {
            showstr(hout, 2 * (j + 1), i + 2, "○", inside_array[i][j].cor, COLOR_HWHITE);
            Sleep(50);
        }

    setcursor(hout, CURSOR_VISIBLE_NORMAL);
```

```
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    gotoxy(hout, 0, row + 3);
}
//draw the color balls with fenge line
void draw3(ball inside_array[][9], int row, int line)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    system("cls");
    setfontsize(hout, L"新宋体", 20);
    gotoxy(hout, 0, 0);
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    cout << "屏幕：" << row * 2 + 5 << "行" << line * 2 + 21 << "列" << endl;

    showstr(hout, 0, 1, "┌", COLOR_HWHITE, COLOR_BLACK);
    for (int i = 0; i < line - 1; i++)
        showstr(hout, 4 * i + 2, 1, "──┬", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 4 * line - 2, 1, "──┐", COLOR_HWHITE, COLOR_BLACK);

    for (int i = 0; i < row - 1; i++)
    {
        showstr(hout, 0, i * 2 + 2, "│", COLOR_HWHITE, COLOR_BLACK);
        for (int j = 0; j < line - 1; j++)
            showstr(hout, 4 * j + 2, i * 2 + 2, "  │", COLOR_HWHITE, COLOR_BLACK);
        showstr(hout, 4 * line - 2, i * 2 + 2, "  │", COLOR_HWHITE, COLOR_BLACK);

        showstr(hout, 0, i * 2 + 3, "├", COLOR_HWHITE, COLOR_BLACK);
        for (int j = 0; j < line - 1; j++)
            showstr(hout, 4 * j + 2, i * 2 + 3, "──┼", COLOR_HWHITE, COLOR_BLACK);
        showstr(hout, 4 * line - 2, i * 2 + 3, "──┤", COLOR_HWHITE, COLOR_BLACK);
    }


    showstr(hout, 0, 2 * row, "│", COLOR_HWHITE, COLOR_BLACK);
    for (int j = 0; j < line - 1; j++)
        showstr(hout, 4 * j + 2, row * 2, "│", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 4 * line - 2, row * 2, "│", COLOR_HWHITE, COLOR_BLACK);

    showstr(hout, 0, 2 * row + 1, "└", COLOR_HWHITE, COLOR_BLACK);
    for (int j = 0; j < line - 1; j++)
        showstr(hout, 4 * j + 2, row * 2 + 1, "──┴", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 4 * line - 2, row * 2 + 1, "──┘", COLOR_HWHITE, COLOR_BLACK);



    for (int i = 0; i < row; i++)
        for (int j = 0; j < line; j++)
            showstr(hout, 2 + j * 4, (i + 1) * 2, "○", inside_array[i][j].cor, COLOR_HWHITE);


    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    gotoxy(hout, 0, row * 2 + 2);

}
//可消除小球标明
void draw4(ball inside_array[][9], int row, int line)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    int flag = 0;
    system("cls");
    setfontsize(hout, L"新宋体", 20);
    gotoxy(hout, 0, 0);
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    cout << "屏幕：" << row * 2 + 5 << "行" << line * 2 + 21 << "列" << endl;

    showstr(hout, 0, 1, "┌", COLOR_HWHITE, COLOR_BLACK);
    for (int i = 0; i < line - 1; i++)
        showstr(hout, 4 * i + 2, 1, "──┬", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 4 * line - 2, 1, "──┐", COLOR_HWHITE, COLOR_BLACK);

    for (int i = 0; i < row - 1; i++)
    {
        showstr(hout, 0, i * 2 + 2, "│", COLOR_HWHITE, COLOR_BLACK);
        for (int j = 0; j < line - 1; j++)
            showstr(hout, 4 * j + 2, i * 2 + 2, "  │", COLOR_HWHITE, COLOR_BLACK);
        showstr(hout, 4 * line - 2, i * 2 + 2, "  │", COLOR_HWHITE, COLOR_BLACK);
```

```
        showstr(hout, 0, i * 2 + 3, "├", COLOR_HWHITE, COLOR_BLACK);
        for (int j = 0; j < line - 1; j++)
            showstr(hout, 4 * j + 2, i * 2 + 3, "──┼", COLOR_HWHITE, COLOR_BLACK);
        showstr(hout, 4 * line - 2, i * 2 + 3, "──┤", COLOR_HWHITE, COLOR_BLACK);
    }


    showstr(hout, 0, 2 * row, "│", COLOR_HWHITE, COLOR_BLACK);
    for (int j = 0; j < line - 1; j++)
        showstr(hout, 4 * j + 2, row * 2, "│", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 4 * line - 2, row * 2, "│", COLOR_HWHITE, COLOR_BLACK);

    showstr(hout, 0, 2 * row + 1, "└", COLOR_HWHITE, COLOR_BLACK);
    for (int j = 0; j < line - 1; j++)
        showstr(hout, 4 * j + 2, row * 2 + 1, "──┴", COLOR_HWHITE, COLOR_BLACK);
    showstr(hout, 4 * line - 2, row * 2 + 1, "──┘", COLOR_HWHITE, COLOR_BLACK);



    for (int i = 0; i < row; i++)
        for (int j = 0; j < line; j++)
        {
            if (inside_array[i][j].dis>0)
            {
                showstr(hout, 2 + j * 4, (i + 1) * 2, "●", inside_array[i][j].cor, COLOR_BLACK);
                flag = 1;
            }
            else
                showstr(hout, 2 + j * 4, (i + 1) * 2, "○", inside_array[i][j].cor, COLOR_HWHITE);
        }
    if (flag == 0)
    {
        gotoxy(hout, 16, 0);
        setcolor(hout, COLOR_BLACK, COLOR_WHITE);
        cout << "未找到初始可消除项";
        return;
    }
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    gotoxy(hout, 0, row * 2 + 2);

}
//消除并附加动画
void dis_act(ball inside_array[][9], int row, int line)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    int i, j;
    for (i = 0; i < row; i++)
        for (j = 0; j < line; j++)
        {
            if (inside_array[i][j].dis>0)
            {
                showstr(hout, 2 + j * 4, (i + 1) * 2, "  ", COLOR_HWHITE, COLOR_HWHITE);
                Sleep(50);//延时0.05秒
                showstr(hout, 2 + j * 4, (i + 1) * 2, "○", inside_array[i][j].cor, COLOR_HWHITE);
                Sleep(50);
                showstr(hout, 2 + j * 4, (i + 1) * 2, "  ", COLOR_HWHITE, COLOR_HWHITE);
            }
        }
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    gotoxy(hout, 0, row * 2 + 2);

}
//落下并填补附加动画
void down_reload(ball inside_array[][9], int row, int line)
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    srand((unsigned int)time(0));
    int i, j;
    for (i = row - 1; i >= 0; i--)
        for (j = line - 1; j >= 0; j--)
            if (inside_array[i][j].cor == 0)
                up_down(inside_array, i, j, i);
    for (i = 0; i < row; i++)
        for (j = 0; j < line; j++)
        {
            if (inside_array[i][j].cor == 0)
            {
                inside_array[i][j].cor = rand() % 9 + 1;
                showstr(hout, 2 + j * 4, (i
```

装 订 线

```
+ 1) * 2, "○", inside_array[i][j].cor,
COLOR_HWHITE);
                    Sleep(100);
                }
            }
    setcolor(hout, COLOR_BLACK,
COLOR_WHITE);
    gotoxy(hout, 0, row * 2 + 2);
}
//标出可移动的球,若不可移动则变回普通形状
void mark_ball(ball inside_array[][9], int
row, int line)
{
    HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
    int i, j;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < line; j++)
        {
            if (inside_array[i][j].cha>0)
                showstr(hout, 2 + j * 4, (i
+ 1) * 2, "◎", inside_array[i][j].cor,
COLOR_HWHITE);
            else
                showstr(hout, 2 + j * 4, (i
+ 1) * 2, "○", inside_array[i][j].cor,
COLOR_HWHITE);
        }
    }
    setcolor(hout, COLOR_BLACK,
COLOR_WHITE);
    gotoxy(hout, 0, row * 2 + 2);
}
//交换两球的位置并出现动画
void ball_change(ball inside_array[][9],
int x, int y, int ex, int ey)
{
    HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
    int temp;
    showstr(hout, 4 * x + 2, 2 * y + 2, "  ",
COLOR_HWHITE, COLOR_HWHITE);
    showstr(hout, 4 * ex + 2, 2 * ey + 2, "
", COLOR_HWHITE, COLOR_HWHITE);
    Sleep(500);
    showstr(hout, 4 * x + 2 + 2 * (ex - x),
2 * y + 2 + ey - y, "○",
inside_array[y][x].cor, COLOR_HWHITE);
    Sleep(500);
    temp = inside_array[y][x].cor;
    inside_array[y][x].cor =
inside_array[ey][ex].cor;
    inside_array[ey][ex].cor = temp;
    if (x - ex != 0)
        showstr(hout, 4 * x + 2 + 2 * (ex -
x), 2 * y + 2 + ey - y, "|", COLOR_HWHITE,
COLOR_BLACK);
    if (y - ey != 0)
        showstr(hout, 4 * x + 2 + 2 * (ex -
x), 2 * y + 2 + ey - y, "—", COLOR_HWHITE,
COLOR_BLACK);
    showstr(hout, 4 * x + 2, 2 * y + 2, "○",
inside_array[y][x].cor, COLOR_HWHITE);
    showstr(hout, 4 * ex + 2, 2 * ey + 2, "○",
inside_array[ey][ex].cor, COLOR_HWHITE);

}
void menu()
{
    cout <<
"*****************************************
*****************************\n"
        << "1.内部数组，生成初始状态，寻找
是否有初始可消除项\n"
        << "2.内部数组，消除初始可消除项后
用非0项下落并用0填充\n"
        << "3.内部数组，消除初始可消除项后
查找消除提示\n"
        << "4.n*n的框架（无分割线），显示
初始状态\n"
        << "5.n*n的框架（有分割线），显示
初始状态\n"
        << "6.n*n的框架（有分割线），显示
初始状态及初始可消除项\n"
        << "7.n*n的框架（有分割线），消除
初始可消除项后显示消除提示\n"
        << "8.cmd图形界面完整版\n"
        << "9.从文件中读取以验证查找消除提
示的算法的正确性\n"
        << "0.退出\n"

        <<
"*****************************************
*****************************\n"
        << "[请选择0-8]";
}
void q1(HANDLE hout, ball inside_array[][9],
int *row, int *line)
{
    int i, j, flag = 0;
    system("cls");
    int_xy(row, line);
    set_ball(inside_array, *row, *line);
    system("cls");
    array_out(inside_array, *row, *line);
    cout << "按回车键寻找可消除项";
    while (_getch() != 13);
```

装 订 线

```
    for (i = 0; i < *row; i++)
        for (j = 0; j < *line; j++)
        {
            if (find_dis(inside_array, i, j, *row, *line) != 0)
            {
                char ch[2] = "\0";
                ch[0] = char(inside_array[i][j].cor + '0');
                showstr(hout, 2 * j + 4, i + 3, ch, COLOR_YELLOW, COLOR_WHITE);
                flag = 1;
            }
        }
    gotoxy(hout, 0, *row + 4);
    if (flag == 0)
        cout << "\n未找到可消除项";
    setcursor(hout, CURSOR_VISIBLE_NORMAL);
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);

}
void q2(HANDLE hout, ball inside_array[][9], int *row, int *line)
{
    int i, j, flag = 0;
    srand((unsigned int)time(0));
    q1(hout, inside_array, row, line);
    cout << "\n按回车键进行消除并下落";
    while (_getch() != 13);
    disappear(inside_array, *row, *line);
    for (i = *row - 1; i >= 0; i--)
        for (j = *line - 1; j >= 0; j--)
            if (inside_array[i][j].cor == 0)
                up_down1(inside_array, i, j, i);
    system("cls");
    array_out(inside_array, *row, *line);
    for (i = 0; i < *row; i++)
        for (j = 0; j < *line; j++)
            if (inside_array[i][j].cor == 0)
                showstr(hout, 2 * j + 4, i + 3, "0", COLOR_YELLOW, COLOR_WHITE);
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    gotoxy(hout, 0, *row + 4);
    cout << "\n按回车键进行随机填补";
    while (_getch() != 13);
    for (i = 0; i < *row; i++)
        for (j = 0; j < *line; j++)
            if (inside_array[i][j].cor == 0)
            {
                inside_array[i][j].cor = rand() % 9 + 1;
                char ch[2] = "\0";
                ch[0] = char(inside_array[i][j].cor + '0');
                showstr(hout, 2 * j + 4, i + 3, ch, COLOR_YELLOW, COLOR_WHITE);
            }
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);
    gotoxy(hout, 0, *row + 4);
}
void q3(HANDLE hout, ball inside_array[][9], int *row, int *line)
{
    int i, j;
    q2(hout, inside_array, row, line);
    cout << "\n按回车键查找可交换的小球";
    while (_getch() != 13);
    system("cls");
    array_out(inside_array, *row, *line);
    reset_dis(inside_array, *row, *line);
    reset_cha(inside_array, *row, *line);
    for (i = 0; i < *row; i++)
        for (j = 0; j < *line; j++)
            find_cha(inside_array, i, j, *row, *line);


    for (i = 0; i < *row; i++)
    {
        for (j = 0; j < *line; j++)
        {
            if (inside_array[i][j].cha == 1)
            {
                char ch[2] = "\0";
                ch[0] = char(inside_array[i][j].cor + '0');
                showstr(hout, 2 * j + 4, i + 3, ch, COLOR_YELLOW, COLOR_WHITE);
            }
        }
    }

    gotoxy(hout, 0, *row + 4);
    setcolor(hout, COLOR_BLACK, COLOR_WHITE);



}
```

```
void q4(HANDLE hout, ball inside_array[][9],
int *row, int *line)
{
    system("cls");
    int_xy(row, line);
    set_ball(inside_array, *row, *line);
    system("cls");
    array_out(inside_array, *row, *line);
    cout << "按回车键显示图形";
    while (_getch() != 13);
    draw2(inside_array, *row, *line);
}
void q5(HANDLE hout, ball inside_array[][9],
int *row, int *line)
{
    system("cls");
    int_xy(row, line);
    set_ball(inside_array, *row, *line);
    system("cls");
    array_out(inside_array, *row, *line);
    cout << "按回车键显示图形";
    while (_getch() != 13);
    draw3(inside_array, *row, *line);
}
void q6(HANDLE hout, ball inside_array[][9],
int *row, int *line)
{
    int i, j;
    system("cls");
    int_xy(row, line);
    set_ball(inside_array, *row, *line);
    system("cls");
    array_out(inside_array, *row, *line);
    for (i = 0; i < *row; i++)
        for (j = 0; j < *line; j++)

            find_dis(inside_array, i, j,
*row, *line);


    cout <<"按回车键显示图形及初始可消除项
";
    while (_getch() != 13);
    draw4(inside_array, *row, *line);
}
void q7(HANDLE hout, ball inside_array[][9],
int *row, int *line)
{
    int i, j, flag = 0;
    system("cls");
    int_xy(row, line);
    set_ball(inside_array, *row, *line);
    system("cls");
    array_out(inside_array, *row, *line);
    cout <<"\n按回车键显示图形及初始可消除
项";
    while (_getch() != 13);
    while (1)
    {
        flag = 0;
        reset_dis(inside_array, *row,
*line);
        for (i = 0; i < *row; i++)
            for (j = 0; j < *line; j++)
            {
                find_dis(inside_array, i,
j, *row, *line);
                if
(inside_array[i][j].dis>0)
                    flag = 1;
            }
        draw4(inside_array, *row, *line);
        if (flag == 0)
            break;
        disappear(inside_array, *row,
*line);
        while (_getch() != 13);
        cout << "\n按回车键消除初始可消除
项";
        dis_act(inside_array, *row,
*line);
        down_reload(inside_array, *row,
*line);
    }
    gotoxy(hout, 0, *row * 2 + 2);
    cout << "\n按回车键显示消除提示";
    while (_getch() != 13);
    for (i = 0; i < *row; i++)
        for (j = 0; j < *line; j++)

            find_cha(inside_array, i, j,
*row, *line);
    mark_ball(inside_array, *row, *line);
}
void q8(HANDLE hout, ball inside_array[][9],
int *row, int *line)
{
    int x, y, ex, ey, flag1 = 0, flag = 0, i,
j, score = 0;
    system("cls");
    int_xy(row, line);
    set_ball(inside_array, *row, *line);
    system("cls");
    array_out(inside_array, *row, *line);
    cout << "按回车键开始游戏";
    while (_getch() != 13);
    draw3(inside_array, *row, *line);
    while (1)
```

```
    {
        flag = 0;
        reset_dis(inside_array, *row,
*line);
        for (i = 0; i < *row; i++)
            for (j = 0; j < *line; j++)
            {
                find_dis(inside_array, i,
j, *row, *line);
                if
(inside_array[i][j].dis>0)
                    flag = 1;
            }

        if (flag == 0)
            break;
        disappear(inside_array, *row,
*line);
        dis_act(inside_array, *row,
*line);
        down_reload(inside_array, *row,
*line);
    }
    gotoxy(hout, 0, *row * 2 + 2);
    for (i = 0; i < *row; i++)
        for (j = 0; j < *line; j++)
            find_cha(inside_array, i, j,
*row, *line);
    mark_ball(inside_array, *row, *line);//
初始小球消除下落并标出,基本同q7
    while (1)
    {
        flag1 = 0;//当有小球可移动时，将
flag1置1，直到无小球可移动，退出循环
        reset_cha(inside_array, *row,
*line);//重置小球是否可移动值
       if (mouse_input(hout, inside_array,
&y, &x, &ey, &ex, *row, *line) == 1)
            break;//加入鼠标,可交换小球位
置
        gotoxy(hout, 0, *row * 2 + 2);
        /*cout << "\n按回车键继续";
        while (_getch() != 13);*/
        while (1)//交换位置后，重置可消除
值，然后判断消除，下落填补，继续消除直到无
小球可消除
        {
            flag = 0;
            reset_dis(inside_array, *row,
*line);
            for (i = 0; i < *row; i++)
                for (j = 0; j < *line; j++)
                {
                    find_dis(inside_array, i, j, *row,
*line);
                    if
(inside_array[i][j].dis>0)
                        flag = 1;
                }
            if (flag == 0)
                break;
            disappear(inside_array, *row,
*line);
            count_score(hout,
inside_array, *row, *line, &score);
            /*while (_getch() != 13);
            cout << "\n按回车键消除初始可
消除项";*/
            dis_act(inside_array, *row,
*line);
            down_reload(inside_array,
*row, *line);
        }
        gotoxy(hout, 0, *row * 2 + 2);
        /*cout << "\n按回车键显示消除提示
";
        while (_getch() != 13);*/
        for (i = 0; i < *row; i++)//判断有
无小球可移动，若可移动则标出
            for (j = 0; j < *line; j++)
            {

                find_cha(inside_array, i,
j, *row, *line);
                if
(inside_array[i][j].cha>0)
                    flag1 = 1;
            }
        mark_ball(inside_array, *row,
*line);
        if (flag1 == 0)
            break;
    }
}
void q9(HANDLE hout, ball inside_array[][9])
{
    ifstream fin, fin1;
    char p[20], p1[20];
    int row, line, i, j, x, y;
    system("cls");
    cout << "请输入学号<输入1651234 对应会
打开1651234.dat>" << endl;
    cin >> p;
    strcpy(p1, p);
    strcat(p, ".dat");

    strcat(p1, ".ans");
```

```
    fin.open(p, ios::in);
    if (fin.is_open() == 0)
    {
        cout << "open failed";
        return;
    }
    fin >> row >> line;
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < line; j++)
        {
            fin >> inside_array[i][j].cor;
        }
    }
    fin.close();
    cout << endl << "原始数组" << endl;
    array_out(inside_array, row, line);
    reset_cha(inside_array, row, line);
    for (i = 0; i < row; i++)
        for (j = 0; j < line; j++)
            find_cha(inside_array, i, j,
row, line);
    cout << "我的算法判断结果：" << endl;
    getxy(hout, x, y);
    array_out(inside_array, row, line);

    for (i = 0; i < row; i++)
    {
        for (j = 0; j < line; j++)
        {
            if (inside_array[i][j].cha ==
1)
            {
                char ch[2] = "\0";
                ch[0] =
char(inside_array[i][j].cor + '0');
                showstr(hout, 2 * j + 4, y
+ i + 3, ch, COLOR_YELLOW, COLOR_WHITE);
            }
        }
    }

    gotoxy(hout, 0, y + row + 3);
    setcolor(hout, COLOR_BLACK,
COLOR_WHITE);
    reset_cha(inside_array, row, line);
    cout << "标准答案" << endl;
    fin1.open(p1, ios::in);
    if (fin1.is_open() == 0)
    {
        cout << "open failed";
        return;
    }
    for (i = 0; i < row; i++)
    {
        for (j = 0; j < line; j++)
        {
            fin1 >>
inside_array[i][j].cor;
            inside_array[i][j].cha =
inside_array[i][j].cor;
            if
(inside_array[i][j].cor>90)
                inside_array[i][j].cor -=
90;

        }
    }
    fin1.close();
    getxy(hout, x, y);
    array_out(inside_array, row, line);

    for (i = 0; i < row; i++)
    {
        for (j = 0; j < line; j++)
        {
            if
(inside_array[i][j].cha>90)
            {
                char ch[2] = "\0";
                ch[0] =
char(inside_array[i][j].cor + '0');
                showstr(hout, 2 * j + 4, y
+ i + 3, ch, COLOR_YELLOW, COLOR_WHITE);
            }
        }
    }

    gotoxy(hout, 0, y + row + 3);
    setcolor(hout, COLOR_BLACK,
COLOR_WHITE);
}
int main()
{
    const HANDLE hout =
GetStdHandle(STD_OUTPUT_HANDLE);
    int row, line, flag = 0;
    ball inside_array[9][9];
    while (1)
    {
        system("cls");
        menu();
        int choice = int(_getch());
        switch (choice)
        {
            case '1':
                q1(hout, inside_array,
```

```
&row, &line);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '2':
                    q2(hout, inside_array,
&row, &line);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '3':
                    q3(hout, inside_array,
&row, &line);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '4':
                    q4(hout, inside_array,
&row, &line);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '5':
                    q5(hout, inside_array,
&row, &line);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '6':
                    q6(hout, inside_array,
&row, &line);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '7':
                    q7(hout, inside_array,
&row, &line);
                    gotoxy(hout, 0, row * 2 +
2);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '8':
                    q8(hout, inside_array,
&row, &line);
                    gotoxy(hout, 0, row * 2 +
2);
                    cout << "已无可移动小球，
游戏结束";
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '9':
                    q9(hout, inside_array);
                    cout << "\n按回车键退出";
                    while (_getch() != 13);
                    continue;
                case '0':
                    break;
                default:
                    continue;
            }
        }
        return 0;
    }
}
```