

同济大学

高级语言程序设计课程实验报告



实验名称: 彩球游戏的设计与实现

学 院: 计算机科学与技术学院

专 业: 计算机科学与技术

学 号:                     

姓 名:                     

完成日期: 2024 年 12 月 26 日

## 1. 汉诺塔综合演示

### 1.1. 题目及基本要求描述

#### 1.1.1. 题目目标

实现与Windows版Color linez类似功能的小游戏，具体分为七个功能（A-G）。

#### 1.1.2. 题目功能要求及相关限制

1. 功能A-以数组形式展示随机生成的5个球（有球的位置有颜色）；
2. 功能B-以数组形式展示随机生成的60%的球，并输入移动指令，寻找对应的移动路径；
3. 功能C-以数组形式进行完整的游戏设计（球的位置用不同颜色标出；消除规则自定义；没有任何空位则游戏结束；一次移动得分则不产生新球）；
4. 功能D-以伪图形界面的方式展示随机生成的5个球（无边框）；
5. 功能E-以伪图形界面的方式展示随机生成的5个球（有边框）；
6. 功能F-以伪图形界面的方式展示随机生成的60%的球，并通过鼠标控制完成一次移动（鼠标左键选择，右键退出；实时显示鼠标指向矩阵的位置；移动过程有完整动画显示）；
7. 功能G-以伪图形界面的方式进行完整的游戏设计；
8. 必须VS2022编译通过；
9. 对应的控制台及分辨率必须按照文档要求调整。

## 2. 整体设计思路

**说明：**本游戏程序的消除规则为：超过五个则消除，消除几个得几分，交叉点不重复计数。

整个程序整体设计思路为（主要围绕F/G功能进行说明）：

首先调用color\_linez\_menu()函数，显示选择功能的界面，等待用户输入对应的功能号；

```
-----
A. 内部数组，随机生成初始5个球
B. 内部数组，随机生成60%的球，寻找移动路径
C. 内部数组，完整版
D. 画出n*n的框架（无分隔线），随机显示5个球
E. 画出n*n的框架（有分隔线），随机显示5个球
F. n*n的框架，60%的球，支持鼠标，完成一次移动
G. cmd图形界面完整版
Q. 退出
-----
[请选择:]
```

（功能对应如上图，方便后续报告内容的理解）

如果输入的为q或Q，则程序直接结束；如果不是q或Q，首先执行void Input(&hang, &lie)函数，输入游戏的行、列数；

后续通过输入功能号的不同，进行各个功能的实现；

接下来主要进行功能F/G的设计思路阐述：

输入行、列后，调用void array\_A(int hang, int lie, int array[][9], int number\_ball)函数，在数组中随机生成对应数量的小球，小球颜色随机；

之后调用void draw\_border\_pro(int array[][9], int hang, int lie)函数，进行初始的小球和游戏格（有边框）的绘制；

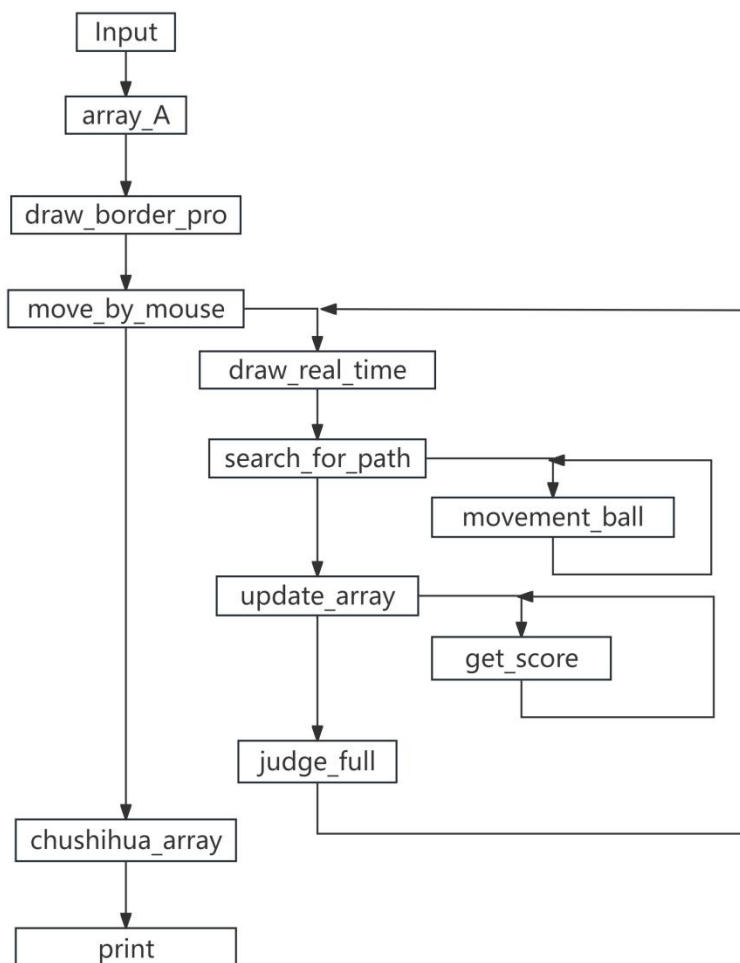
然后调用void move\_by\_mouse(int hang, int lie, int array[][9], int gongnenghao)函数，用户可以通过鼠标来移动小球；

最后调用void chushihua\_array(int array[][9], int hang, int lie)（初始化数组）和void print(void)（输入end以进入下一小题）函数，完成下次功能选用的衔接。

以上是main函数通过调用这些函数实现F/G功能，事实上，上述所提到的函数存在调用其他函数的情况。具体函数之间的相互调用关系将在后续以流程框图的形式展现。

## 3. 主要功能的实现

整个程序的功能实现描述已在整体设计思路中给出，此部分将以流程图的形式展示F/G功能的实现过程（即函数之间的相互调用关系）。



上图是F/G功能的实现逻辑，现对其中涉及的函数进行功能说明：

（函数命名采用中英混合的方式，阅读时注意切换理解）

`void Input(int* hang, int* lie):` 输入行、列；

`void array_A(int hang, int lie, int array[][9], int number_ball):` 在数组内生成对应数量的小球；

`void draw_border_pro(int array[][9], int hang, int lie):` 绘制有分割线的边框；

`void move_by_mouse(int hang, int lie, int array[][9], int gongnenghao):` 鼠标控制函数；

`void draw_real_time(int ball[], int array[][9], int hang, int lie):` 绘制得分框、后续三个彩球颜色的预测框、统计各个彩球数量的统计框；

`void search_for_path(int array[][9], int src_x, int src_y, int dst_x, int dst_y, int hang, int lie, int visited[][9], int* result, int gongnenghao, int x, int y):` 路径搜索函数；

`void movement_ball(int array[][9], int src_x, int src_y, int dst_x, int dst_y, int result, int gongnenghao, int _x, int _y):` 移动小球的动画效果；

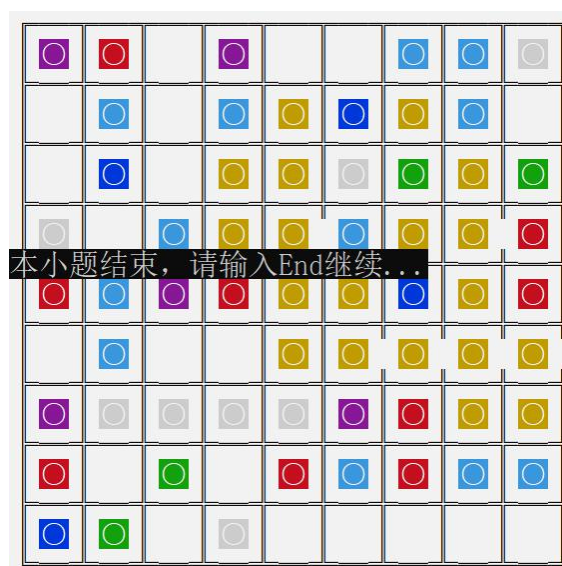
`void update_array(int array[][9], int& flag_y, int& flag_x, int& flag_first, int& flag_second, int& array_src_x, int& array_src_y, int& array_dst_x, int& array_dst_y, int hang, int lie, int ball[], int *sum_score):` 更新图形化界面各个统计值；

`int get_score(int array[][9], int* sum_score, int hang, int lie, int ball[], int gongnenghao):` 移动一步后进行得分统计；

`bool judge_full(int array[][9], int hang, int lie):` 判断游戏格是否填满，填满则游戏结束；

## 4. 调试过程碰到的问题

Question 1描述:在进行小球移动的动画演示时，演示的移动路径总有问题（具体如下图所示）



移动后的轨迹清除不了

**解决方案 1:** 通过对递归函数每次递归一次都设置断点，最终查出是由于递归搜索的时候存在回溯，而函数中没有处理回溯后的轨迹，因此移动轨迹不断重合，消除不了。后来修改了递归的移动逻辑：

```
if (array[src_x + 1][src_y] == 0 && src_x + 1 < hang && visited[src_x + 1][src_y] == 0) { //向下搜索g=9
    movement_ball(array, src_x, src_y, src_x+1, src_y, *result, gongnenghao, x, y); //!!!
    search_for_path(array, src_x + 1, src_y, dst_x, dst_y, hang, lie, visited, result, gongnenghao, x, y);
    movement_ball(array, src_x+1, src_y, src_x, src_y, *result, gongnenghao+1, x, y); // 可能的回溯
}
```

以向下递归有回溯的递归处理为例，一开始movement移动是向下移动，但当search\_for\_path函数递归执行完之后，说明这样递归下去没法到达目的坐标，因此回溯回来后要向上移动（图中gongnenghao=9代表向下移动，gongnenghao=10代表向上移动）。其他向上、向左、向右移动后回溯的逻辑与此相同，回溯回来后总是向相反的方向移动回去。

通过此问题的解决，有助于我更好地理解递归的执行过程。

**Question 2描述:** 在处理移动动画效果时，移动路径的处理还是有问题。类似上图中的，消除移动前的图形，但是却把边框也消除了，或者是在空白格的地方打印边框。

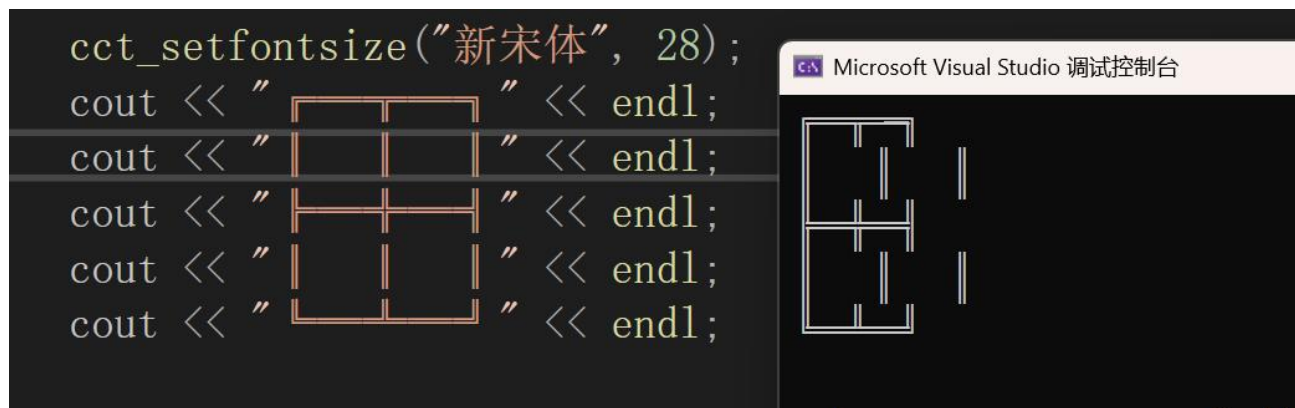
**解决方案 2:** 由于中文制表符在打印时，横向x占用2个字节的位置，纵向y还是占用一个字节的位置，因此横向左右移动和纵向的上下移动的处理逻辑是不一样的。出错就是因为将横向和纵向等同处理了，导致移动路径没有处理好。

```
else if (gongnenghao == 11) {
    for (int x = 4 * src_y + 2; x <= 4 * dst_y + 2; x++) { //右移
        int y = 2 * src_x + 1;
        if (x % 2 == 0) {
            if (x % 4 == 0) {
                cct_showstr(x - 2, y, " ", COLOR_HWHITE, COLOR_HWHITE, 1, -1);
            }
            else {
                cct_showstr(x - 2, y, " | ", COLOR_HWHITE, COLOR_BLACK, 1, -1);
            }
        }
        else {
            cct_showstr(x - 2, y, " ", COLOR_HWHITE, COLOR_HWHITE, 1, -1);
        }
        cct_showstr(x, y, "O", array[_x][_y], COLOR_HWHITE, 1, -1);
        Sleep(100);
    }
}
```

因此，如上图所示，对横向处理逻辑增加一个条件即可。

**Question 3描述:** 打印中文制表符时出现了如下图所示的问题，直接打印中文制表符会出现重合然后打印混乱的情况；(图片来自群里同学的提问，由于本人遇到类似问题时忘记截图了，故借来一用)





```

cct_setfontsize("新宋体", 28);
cout << "  " << endl;
cout << "  " << endl;
cout << "  " << endl;
cout << "  " << endl;
cout << "  " << endl;
    
```

The image shows a screenshot of a C++ program in Visual Studio. The code defines a font size and prints five lines of two spaces each. The output on the right shows a 2x2 grid of Chinese characters (新宋体) and its corresponding ASCII art representation.

**解决方案 3:** 我觉得是因为在打印的时候，光标处打印了一个占用两字节的字符，但是下次打印前光标只会向后移动一个字节的位置，因此两次打印的中文制表符便会有重叠的情况。直接用cct\_showstr函数可以解决。但我是做完弹出式菜单的题目后才看到群里老师的解答，因此我就没有用cct\_showstr函数，而是选择了“手动”校准的方式，就是每打印完一个中文制表符，就再打印一个空格，让光标移动到正确的位置，然后再打印下一个中文制表符，这样就不会有重叠的情况出现。但是“手动”校准会让程序多很多冗余代码（但是我弹出式菜单写完后真的真的懒得改成cct\_Showstr的方法了、、、），因此在彩球作业中便采用了cct\_shoestr的方式，确实简单很多。

**其他Question描述:** 其实在调试过程中遇到了很多问题，但基本上都是由于处理逻辑不当所导致。具体经验在后续的心得体会中会说到。

## 5. 心得体会

### 5.1. 完成本次作业得到的一些心得体会，经验教训

本次作业没有先前的作业做铺垫，一切都是自己从头开始设计。有了之前大作业的经验积累，此次作业还是将每一个要实现的功能划分为不同的功能函数去实现，然后通过绘制流程图，理解函数之间的相互调用关系，不断修改函数传递的参数。

然后本次题目在打印边框时其实涉及到了光标的不断移动，这就要求在编写程序的时候要非常熟悉每一步完成后光标在哪个位置，这样才能正确打印出来。但这个过程非常冗长，如果中断一次，下次再开始的时候就会忘掉该从哪里开始，就又得从头去理解，会浪费很多时间。基于这个情况，每次写大作业的时候我都会在纸上画出每一步光标会移动到哪里，下次再写的时候看到上次画的图就能很快地回忆起来。但功能G实现的时候我还是选择直接写死得分框、预测框、统计框的开始绘制坐标（毕竟考虑完所有情况后写死一个不会出错的坐标真的非常方便，程序在逻辑上也会简单很多）。

最后是函数处理逻辑方面。鉴于程序本身实现有先后，所以有些功能必须要考虑先后问题，不然最后的呈现就会有问题。比如此次彩球作业功能G的实现，由于时序错乱，导致预测框展示的是此次新加的三个球，并不是下次要添加的球。所以得先执行生成三个彩球并展示到界面的函数，再执行鼠标移动、更新

数据的函数。

## 5.2. 通过综合题 1/2 中有关函数的分解与使用，总结你在完成过程中是否考虑了前后小题的关联关系，是否能尽可能做到后面小题有效利用前面小题已完成的代码，如何才能更好地重用代码？

其实我感觉我个人在前后小题逻辑的贯通上是有一定问题的。每次解决一个多功能的复杂程序时，我都选择先不去考虑后续如何，先按照自己的想法把该功能实现，然后各个功能都实现后，再放在一起重新思考各个功能之间的逻辑能否整合。这样虽然最后也能很好地重用代码，但是会特别浪费时间。但我觉得个人的思维模式很难改变，因此目前我也不知道这种问题如何解决。

但是最后综合功能的实现一定是要和前面已实现的一些小功能联系起来的，我个人觉得实现得还有待提高。因为有些函数得通用性不是很好，这个功能能用，下个功能就得大改。经过我的检查，我觉得是因为我在写每个函数的时候里面有太多“写死”的量，因此下一个功能实现“写死”的量改变了，这个函数就不能复用了。基于这种情况，就是在下次写函数的时候，尽量函数中不要出现任何“写死”的量，尽量都用函数传递的参数去表示，这样代码的重用率就会高很多。

## 6. 附件：源程序

```
void Input(int* hang, int* lie);
void print(void);
void print_array(int hang, int lie, int array[][9], int visited[][9], int gongnenghao);
void array_A(int hang, int lie, int array[][9], int number_ball);
void chushihua_array(int array[][9], int hang, int lie);
void input_move_instruction(int src, int dst, int array[][9], int hang, int lie, int gongner);
void judge_input(char str[], int hang, int lie, int x, int y);
void search_for_path(int array[][9], int src_x, int src_y, int dst_x, int dst_y, int hang, int lie);
void generation_of_3_ball(int ball[]);
void change_array_and_print(int src_x, int src_y, int dst_x, int dst_y, int array[][9], int ball[], int gongnenghao);
int get_score(int array[][9], int* sum_score, int hang, int lie, int ball[], int gongnenghao);
bool judge_full(int array[][9], int hang, int lie);
void draw_border_pause();
void draw_border(int array[][9], int hang, int lie);
void draw_border_pro(int array[][9], int hang, int lie);
void move_by_mouse(int hang, int lie, int array[][9], int gongnenghao);
void movement_ball(int array[][9], int src_x, int src_y, int dst_x, int dst_y, int result, int i);
void update_array(int array[][9], int& flag_y, int& flag_x, int& flag_first, int& flag_secord);
void draw_real_time(int ball[], int array[][9], int hang, int lie);
void draw_score();
void draw_sort(int array[][9], int hang, int lie);
int cal_number(int array[][9], int hang, int lie, int color);
```

本部分将给出上图框起来的函数源程序，未给出的为常规输入处理或输出文字的常规内容（或由于源程序含量不得大于50%，故不再展示）。

```

void search_for_path(int array[][9], int src_x, int src_y, int dst_x, int dst_y, int hang, int lie,
int visited[][9], int* result, int gongnenghao, int x, int y)
{
    visited[src_x][src_y] = int('*' - '0');//表示已经访问过
    if (src_x == dst_x && src_y == dst_y) {
        *result = 1;
        if (gongnenghao == 2) {
            cout << "查找结果数组: " << endl;//cout << "找到了";
            print_array(hang, lie, visited, visited, 2);
            cout << "移动路径(不同色标识): " << endl;
            print_array(hang, lie, array, visited, 3);
        }
        else if(gongnenghao==3){
            cout << "移动后的数组(不同色标识): " << endl;
        }
        return;
    }
    if (array[src_x + 1][src_y] == 0 && src_x + 1 < hang && visited[src_x + 1][src_y] == 0) { //向下搜索 g=9
        movement_ball(array, src_x, src_y, src_x+1, src_y, *result, gongnenghao, x, y);
        search_for_path(array, src_x + 1, src_y, dst_x, dst_y, hang, lie, visited, result,
gongnenghao, x, y);
        movement_ball(array, src_x+1, src_y, src_x, src_y, *result, gongnenghao+1, x, y);
    }
    if (array[src_x - 1][src_y] == 0 && src_x - 1 >= 0 && visited[src_x - 1][src_y] == 0) { //向上搜索 g=10
        movement_ball(array, src_x, src_y, src_x - 1, src_y, *result, gongnenghao+1, x, y);
        search_for_path(array, src_x - 1, src_y, dst_x, dst_y, hang, lie, visited, result,
gongnenghao, x, y);
        movement_ball(array, src_x-1, src_y, src_x, src_y, *result, gongnenghao, x, y);
    }
    if (array[src_x][src_y + 1] == 0 && src_y + 1 < lie && visited[src_x][src_y + 1] == 0) { //向右搜索 g=11
        movement_ball(array, src_x, src_y, src_x, src_y+1, *result, gongnenghao+2, x, y);
        search_for_path(array, src_x, src_y + 1, dst_x, dst_y, hang, lie, visited, result,
gongnenghao, x, y);
        movement_ball(array, src_x, src_y+1, src_x, src_y, *result, gongnenghao+3, x, y);
    }
    if (array[src_x][src_y - 1] == 0 && src_y - 1 >= 0 && visited[src_x][src_y - 1] == 0) { //向左搜索 g=12
        movement_ball(array, src_x, src_y, src_x, src_y - 1, *result, gongnenghao+3, x, y);
        search_for_path(array, src_x, src_y - 1, dst_x, dst_y, hang, lie, visited, result,
gongnenghao, x, y);
        movement_ball(array, src_x, src_y-1, src_x, src_y, *result, gongnenghao+2, x, y);
    }
    if (src_x == x && src_y == y && *result == 0) {
        int tmpx, tmpy;
        cct_getxy(tmpx, tmpy);
        cct_gotoxy(0, tmpy + 1);
        cout << "找不到相关路径" << endl;
    }
    return;
}

```



```

}

int get_score(int array[][9], int* sum_score, int hang, int lie, int ball[], int gongnenghao)
{
    /*得分规则：只有横着，竖着超过五个才可消除得分，消除几个得几分哈*/
    int number1[8] = { 0 };
    for (int i = 0; i < 8; ++i) {
        number1[i] = cal_number(array, hang, lie, i);
    }
    int k = 0, m = 0, x = 0, y = 0, tmp = 0;;
    int tmp_score = 0, count = 0;
    for (int i = 0; i < hang; ++i) {
        for (int j = 0; j < lie; j++) {
            if (array[i][j] != 0) {
                //横着找可以消除的
                for (k = j; k >= 0; --k) {
                    if (array[i][k] != array[i][j]) {
                        k++;
                        break;
                    }
                    ++count;
                }
                for (m = j + 1; m < lie; ++m) {
                    if (array[i][m] != array[i][j]) {
                        m--;
                        break;
                    }
                    ++count;
                }
                if (count >= 5) {
                    tmp_score = tmp_score + count;
                    for (int q = k; q <= m; ++q) {
                        array[i][q] = 0;
                        if (gongnenghao == 8) {
                            cct_showstr(4 * q + 2, 2 * i + 1, " ", COLOR_HWHITE, COLOR_HWHITE,
1, -1);
                        }
                    }
                    count = 0;
                }
            }
            else {
                count = 0;
                //竖着找可以消除的
                for (k = i; k >= 0; --k) {
                    if (array[k][j] != array[i][j]) {
                        k++;
                        break;
                    }
                    ++count;
                }
                for (m = i + 1; m < hang; ++m) {
                    if (array[m][j] != array[i][j]) {
                        m--;
                        break;
                    }
                }
            }
        }
    }
}

```

```

    }
    ++count;
}
if (count >= 5) {
    tmp_score = tmp_score + count;
    for (int q = k; q <= m; ++q) {
        array[q][j] = 0;
        if (gongnenghao == 8) {
            cct_showstr(4 * j + 2, 2 * q + 1, " ", COLOR_HWHITE, COLOR_HWHITE,
1, -1);
        }
    }
    count = 0;
}
else {
    count = 0;
}
}
}
}
*sum_score = *sum_score + tmp_score;
if (gongnenghao == 8 && tmp_score != 0) {
    cct_showstr(47, 9, "○", COLOR_HWHITE, COLOR_HWHITE, 1, -1);
    cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
    cout << ':' << setw(2) << cal_number(array, hang, lie, 0) << "/" << fixed << setprecision(2)
<< double(cal_number(array, hang, lie, 0)) / hang / lie * 100 << "%" << " 消除-0";
    for (int i = 1; i < 8; ++i) {
        cct_showstr(47, 9 + i, "○", i, COLOR_HWHITE, 1, -1);
        cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
        int x, y;
        cct_getxy(x, y);
        cct_gotoxy(x, y);
        cout << ':' << setw(2) << cal_number(array, hang, lie, i) << "/" << fixed <<
setprecision(2) << double(cal_number(array, hang, lie, i)) / hang / lie * 100 << "%" << " 消除-" << setw(2)
<< number1[i] - cal_number(array, hang, lie, i) << " ";
    }
}
if (tmp_score == 0) {
    while (tmp < 3 && !judge_full(array, hang, lie)) {
        x = int(fabs(rand() % hang));
        y = int(fabs(rand() % lie));
        if (array[x][y] == 0) {
            array[x][y] = ball[tmp];
            if (gongnenghao == 8) {
                cct_showstr(4 * y + 2, 2 * x + 1, "○", array[x][y], COLOR_HWHITE, 1, -1);
            }
            ++tmp;
        }
    }
}
return tmp_score;
}

```

```

void move_by_mouse(int hang, int lie, int array[][9], int gongnenghao)
{
    cct_setcursor(CURSOR_INVISIBLE);
    int x, y, flag_y = 0, flag_x = 0, flag_first = 0, flag_second = 0, array_src_x = 0, array_src_y
= 0, array_dst_x = 0, array_dst_y = 0, sum_score=0;
    cct_getxy(x, y);
    cct_gotoxy(0, y);
    cout << "鼠标移动, 左键单击选择, 右键单击退出" << endl;
    cct_enable_mouse();
    cout << "[当前光标] ";
    cct_getxy(x, y);
    int MX, MY, MAction, keycode1, keycode2, state = CCT_MOUSE_EVENT, ball[3] = { 0 };
    if (gongnenghao == 2) { //画得分框、统计框、和预告彩球 //cct_gotoxy(45, 0); //cct_gotoxy(0, 22);
        for (int i = 0; i < 3; ++i) {
            ball[i] = rand() % 7 + 1;
        }
        draw_real_time(ball, array, hang, lie);
    }
    while (1) {
        cct_gotoxy(x, y);
        state = cct_read_keyboard_and_mouse(MX, MY, MAction, keycode1, keycode2);
        for (int i = 1; i <= hang; ++i) {
            if (MY == 2 * i - 1) {
                flag_y = i;
            }
        }
        for (int i = 1; i <= lie; ++i) {
            if (MX == 4 * i - 2 || MX == 4 * i - 1) {
                flag_x = i;
            }
        }
        if (flag_y == 0 || flag_x == 0) {
            cout << "位置非法";
        }
        else {
            cout << char('A' + flag_y-1) << " 行" << flag_x << " 列";
            if (MAction == MOUSE_LEFT_BUTTON_CLICK && array[flag_y - 1][flag_x -
1] != 0 && flag_first == 0) {
                cct_showstr(4 * flag_x-2, 2*flag_y-1, "★", array[flag_y - 1][flag_x - 1],
COLOR_HWHITE, 1, -1); //选中了就不能再选其他的!!!
                cct_setcolor(COLOR_BLACK, COLOR_WHITE);
                flag_first = 1;
                array_src_x = flag_y - 1;
                array_src_y = flag_x - 1;
            }
            else if (MAction == MOUSE_LEFT_BUTTON_CLICK && array[flag_y - 1][flag_x - 1] == 0 &&
flag_second == 0) {
                flag_second = 1;
                array_dst_x = flag_y - 1;
                array_dst_y = flag_x - 1;
            }
        }
        if (MAction == MOUSE_RIGHT_BUTTON_CLICK) {
            break;
        }
    }
}

```

```

    }
    flag_x = 0;
    flag_y = 0;
    if (flag_first == 1 && flag_second == 1) {
        int visited[9][9] = { 0 }, result = 0;
        search_for_path(array, array_src_x, array_src_y, array_dst_x, array_dst_y, hang, lie,
visited, &result, 4, array_src_x, array_src_y);
        if (result == 1) {
            int visited_1[9][9] = { 0 }, result_1 = 0;
            search_for_path(array, array_src_x, array_src_y, array_dst_x, array_dst_y, hang,
lie, visited_1, &result_1, 9, array_src_x, array_src_y); //起始 array[str[0] - 'A'][str[1] - '0' - 1]
            if (gongnenghao == 2) {
                update_array(array, flag_y, flag_x, flag_first, flag_second, array_src_x,
array_src_y, array_dst_x, array_dst_y, hang, lie, ball, &sum_score);
                if (judge_full(array, hang, lie)) {
                    cct_gotoxy(0, 21);
                    break;
                }
            }
        }
        else {
            if (gongnenghao == 2) {
                break;
            }
        }
        if (gongnenghao == 1) {
            break;
        }
    }
}
cct_setcursor(CURSOR_VISIBLE_NORMAL);
cct_disable_mouse();
}

```