实验报告

|  |  |
| --- | --- |
| 实验名称 | 彩球游戏的设计与实现 |
| 班 级 | 计算机1班 |
| 学 号 | 1752762 |
| 姓 名 | 魏鹳达 |
| 完成日期 | 2018.12.30 |

1. **彩球游戏基本要求及游戏规则**
   1. **游戏规则**

游戏区域为7-9行，7-9列，共有7种颜色的彩球随机出现。

玩家可以移动每一颗彩球（前提是路径相通），每次移动随机在游戏区域中随机生成三颗彩球，出现彩球的颜色提前预示给玩家，若在横向、纵向或斜向有五个或以上相连的同颜色球，便消去相连彩球获得相应得分且不会出现新的彩球。

游戏进行到彩球填满游戏区域使其为死局时结束。

* 1. **文字版游戏**

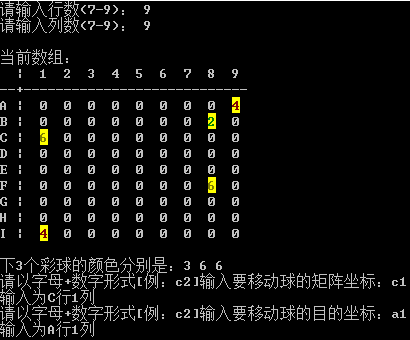
输入行数和列数。

在规定范围内随机生成五个彩球的位置，然后打印整个内部数组（以不同颜色的方式显示彩球）。

输入要移动球的位置及目标位置，若输如非法则给出错误提示并重新输入。

输入完成后，判断是否能到达目标位置，若能则移动彩球，根据是否有消除的彩球判断是否产生新的彩球，若不能移动彩球则再次重新输入。

若在规定范围内无位置可以生成球则游戏结束。

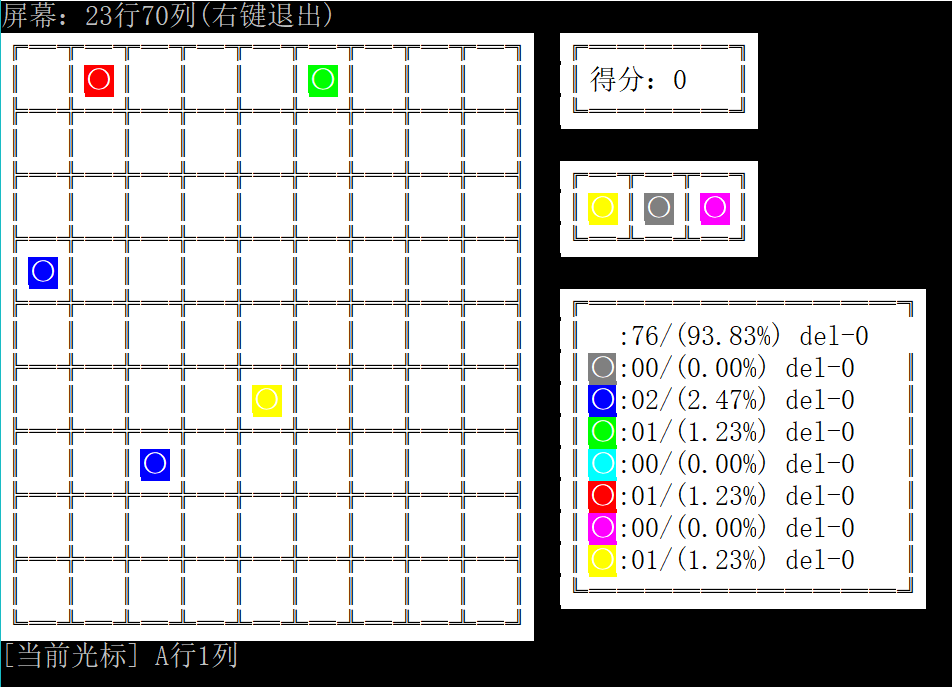
下图为示例。

* 1. **图形版游戏**

在1.2的基本要求上改为以伪图形化的方式打印输出。

移动方式改为可用鼠标和键盘两种方式在图形的基础上进行，额外输出当前得分及彩球分布情况统计。

此外，彩球移动时要求以动画形式进行，游戏进行时可以按右键退出。

下图为示例。

1. **整体设计思路**

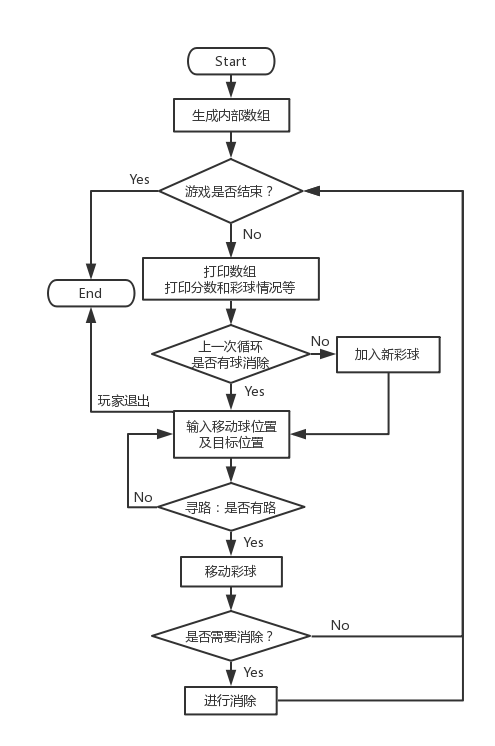
避免使用全局变量，需要的变量通过函数的参数来传递，将每个功能重复使用的代码实现为函数，尽量以共用函数的方式来减少冗余代码。

整个工程的代码大致分为内部数组操作部分函数、打印内部数组信息部分函数及工具部分函数。通过将一个大问题分解为若干个子问题，子问题再分解为小问题的方式细化代码分工，便于主要功能实现的同时也提高了代码共用量。

内部数组操作部分大致分为：内部数组生成、彩球移动、彩球添加、寻路、判断游戏是否结束、判断是否需要消除六个部分。其中寻路函数通过递归的方式求解，缩小了问题的规模。

打印内部数组信息部分大致分为：打印彩球、移动彩球（打印路径）、打印寻路结果、打印得分及彩球情况四个部分。

工具部分为需要反复使用的函数，这里就不再赘述。

1. **主要功能的实现**
2. **调试过程碰到的问题**
   1. **引用**

一些变量需要重复使用并且需要运用到不同函数中，而且需要在每次单独的游戏前置为一个统一的值，在不允许使用全局变量的情况下，就需要使用引用这一方式来辅助完成。

* 1. **算法求解**

寻路函数实现时，需要与现有内部数组结合考虑，并且需要避免所有可能会出现死循环的写法，例如控制不走已走过路径。实际实现时，这一部分使用的是暴力求解的方式，在效率和结果上会差于理想路线，有很大优化的空间。

* 1. **图形控制**

第三，在实现鼠标和键盘移动的方式时，需要考虑有的坐标值是无效值，如指针在边框线、游戏区域外等。图形打印相关动画时，对坐标的把握需要精确无误，同时要考虑到不同行列数时的不同情况。

1. **心得体会**
   1. **经验教训**

一个综合性的程序是若干个部分的结合，实现这样的程序时，需要在实现每个小部分时考虑到其他部分可能的再次调用的关系，并对函数做出调整。同时将需要重复使用的代码片段以函数的方式实现，减少代码重复量，提高可读性。

本次实验吸取了上次实验的教训，在开始前就做了整体计划，有了计划后，一小步一小步地实现某个功能就简单不少了。但还是难免有小部分疏漏，对函数考虑前后使用的情况考虑仍有不周的情况。针对这一情况，采取了有默认参数形参的函数形式，效率上提高了很多。

实现时尽量把每个函数分开，这样一来是代码的清晰，二来是便于以后不管是代码有错，还是可以进行优化，操作起来都会简单很多，而不必大刀阔斧地修改。

希望能够吸取经验教训，在做综合性的程序时依旧需要多考虑整体中各个部分的联系，同时也要多学习一些方法，以备意外情况的发生而需要改动函数来作为后路。

* 1. **前后小题联系**

前后小题的关联程度较高，后一步的功能常常需要调用之前的宫男，但完成过程中由于思考深度不够，一些细节还是处理得不到位，导致仍有部分代码重复，有待简化。

在做综合性的程序时，需要多进行整体上的思考，将整体与部分联系起来，“牵一发而动全身”，以使得尽可能多地重用代码，提高效率的同时也使得代码简洁易读。

1. **附件：源程序**

//90-b2-main.cpp

#include <iostream>

#include <conio.h>

#include <time.h>

#include <stdlib.h>

#include "cmd\_console\_tools.h"

#include "90-b2.h"

using namespace std;

int menu()

{

cout << "---------------------------------------------" << endl;

cout << "1.内部数组，随机生成初始5个球" << endl;

cout << "2.内部数组，随机生成60%的球，寻找移动路径" << endl;

cout << "3.内部数组，完整版" << endl;

cout << "4.画出n\*n的框架（无分隔线），随机显示5个球" << endl;

cout << "5.画出n\*n的框架（有分隔线），随机显示5个球" << endl;

cout << "6.n\*n的框架，60%的球，支持鼠标，完成一次移动" << endl;

cout << "7.cmd图形界面完整版" << endl;

cout << "8.cmd图形界面完整版 - 支持同时读键（额外加分）" << endl;

cout << "0.退出" << endl;

cout << "---------------------------------------------" << endl;

cout << "[请选择0 - 8] ";

int ch;

while (1)

{

ch = \_getch();

if (ch >= '0' && ch <= '8')

{

cout << ch - '0' << endl;

to\_be\_continued(NULL);

break;

}

}

return ch;

}

int main()

{

srand((unsigned)time(0));

int arr[MAX\_ROW + 2][MAX\_COL + 2], row, col, ball[3];

int path[MAX\_ROW + 2][MAX\_COL + 2] = { 0 };

int ball\_conut[8] = { 0 };

setcursor(CURSOR\_INVISIBLE);

while (1)

{

setconsoleborder(80, 25, 80, 1000);

setfontsize("Terminal", 16, 8);

for (int i = 1; i < 8; i++)

ball\_conut[i] = 0;

int choice = menu();

if (choice == '0')

return 0;

cls();

input\_int("请输入行数(7-9)： ", &row, 7, 9, 0, 0);

input\_int("请输入列数(7-9)： ", &col, 7, 9, 0, 1);

switch (choice)

{

case '1':

array\_create(arr, row, col, 5);

array\_print\_text("初始数组：", arr, row, col);

break;

case '2':

array\_create(arr, row, col, int(row\*col\*0.6));

array\_print\_text("当前数组：", arr, row, col);

ball\_creat\_print(ball, 0);

if (array\_print\_move\_text(arr, path, row, col))

{

path\_print\_result\_text(path, row, col);

path\_print\_move\_text(arr, path, row, col);

}

break;

case '3':

array\_create(arr, row, col, 5);

while (!is\_over(arr, row, col))

{

static int ball\_flag = 0;

array\_print\_text("当前数组：", arr, row, col);

ball\_creat\_print(ball, ball\_flag);

int t = array\_print\_move\_text(arr, path, row, col, 1);

if (t == 1)

{

for (int i = 0; i < 3; i++)

{

array\_add(arr, row, col, ball[i]);

if (is\_over)

i = 3;

}

ball\_flag = 0;

}

else

ball\_flag = 1;

if (t)

array\_print\_text("移动后数组：", arr, row, col);

}

break;

case '4':

array\_create(arr, row, col, 5);

array\_print\_text("初始数组：", arr, row, col);

to\_be\_continued(NULL);

setconsoleborder(26 + col, 15, 26 + col, 15);

setfontsize("新宋体", 28);

cout << "屏幕：15行" << 26 + col << "列" << endl;

array\_print\_img\_no(arr, row, col);

break;

case '5':

array\_create(arr, row, col, 5);

array\_print\_text("初始数组：", arr, row, col);

to\_be\_continued(NULL);

setconsoleborder(4 \* col + 3, 2 \* row + 5, 4 \* col + 2, 2 \* row + 5);

setfontsize("新宋体", 28);

cout << "屏幕：" << 2 \* row + 5 << "行" << 4 \* col + 2 << "列" << endl;

array\_print\_img\_yes(arr, row, col);

break;

case '6':

array\_create(arr, row, col, int(row\*col\*0.6));

setconsoleborder(4 \* col + 3, 2 \* row + 5, 4 \* col + 2, 2 \* row + 5);

setfontsize("新宋体", 28);

cout << "屏幕：" << 2 \* row + 5 << "行" << 4 \* col + 2 << "列（右键退出）" << endl;

array\_print\_img\_yes(arr, row, col);

array\_print\_move\_img(arr, path, row, col);

break;

case '7':

case '8':

array\_create(arr, row, col, 5);

setconsoleborder(4 \* 9 + 32, 2 \* row + 5, 4 \* 9 + 2, 2 \* row + 5);

setfontsize("新宋体", 28);

while (!is\_over(arr, row, col))

{

static int ball\_flag = 0;

cls();

cout << "屏幕：" << 2 \* row + 5 << "行" << 4 \* col + 2 << "列（右键退出）" << endl;

array\_print\_img\_yes(arr, row, col);

ball\_creat\_print\_img(ball, ball\_flag);

score\_print(ball\_conut);

ball\_count\_print(arr, ball\_conut, row, col);

int t = array\_print\_move\_img(arr, path, row, col, ball\_conut, choice - '7');

if (t == 1)

{

for (int i = 0; i < 3; i++)

{

array\_add(arr, row, col, ball[i]);

if (is\_over)

i = 3;

}

ball\_flag = 0;

}

else if (t == -1)

break;

else

ball\_flag = 1;

}

break;

default:

cout << "error" << endl;

break;

}

to\_be\_continued("本小题结束");

}

return 0;

}

//90-b2.h

#define MAX\_ROW 9

#define MAX\_COL 9

//内部数组生成

void array\_create(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col, int n\_balls);

//添加球

void array\_add(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col, const int ball);

//移动球

void array\_move(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row\_from, const int col\_from, const int row\_to, const int col\_to);

//寻路 有路返回1 无路返回0 数组path中经过的路径为1，其余为0

int path\_find(const int arr[MAX\_ROW + 2][MAX\_COL + 2], int path[MAX\_ROW + 2][MAX\_COL + 2], const int row\_from, const int col\_from, const int row\_to, const int col\_to, int &flag\_find);

//判断游戏是否结束

int is\_over(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col);

//判断是否消除球并进行相应操作 消除返回1 否则返回0

int remove(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row\_target, const int col\_target, const int row, const int col, int \*ball\_count = NULL);

//文本形式打印内部数组

void array\_print\_text(const char\* s, const int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col, const int color = 1);

//文本形式移动内部数组 未移动返回0 移动但未消除返回1 移动且有消除返回2

int array\_print\_move\_text(int arr[MAX\_ROW + 2][MAX\_COL + 2], int path[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col, const int move\_flag = 0);

//文本打印路径查找结果

void path\_print\_result\_text(const int path[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col);

void path\_print\_move\_text(const int arr[MAX\_ROW + 2][MAX\_COL + 2], const int path[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col);

//图形打印内部数组（无分隔线）

void array\_print\_img\_no(const int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col);

//图形打印内部数组（有分隔线）

void array\_print\_img\_yes(const int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col);

//图形移动内部数组 移动但未消除返回1 移动且有消除返回2

int array\_print\_move\_img(int arr[MAX\_ROW + 2][MAX\_COL + 2], int path[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col, int \*ball\_count = NULL, const int can\_keyboard = 0);

//暂停继续 显示信息

void to\_be\_continued(const char \*prompt);

//输入数据

void input\_int(const char \*s, int \*target, const int low, const int high, const int x, const int y);

void input\_str2(const char \*s, char \*target, const int row\_low, const int row\_high, const int col\_low, const int col\_high);

//生成球并打印信息

void ball\_creat\_print(int ball[3], const int again\_flag);

void ball\_creat\_print\_img(int ball[3], const int again\_flag);

//图形界面 打印得分

void score\_print(const int ball\_conut[8]);

//图形界面 打印彩球情况

void ball\_count\_print(const int arr[MAX\_ROW + 2][MAX\_COL + 2], const int removeball\_conut[8], const int row, const int col);

//子菜单7所用部分函数

void array\_create(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col, int n\_balls)

{

int i, j;

//初始化 边界为-1 内部全为0

for (i = 0; i <= col + 1; i++)

{

arr[0][i] = -1;

arr[row + 1][i] = -1;

}

for (i = 1; i <= row; i++)

{

arr[i][0] = -1;

arr[i][col + 1] = -1;

for (j = 1; j <= col; j++)

arr[i][j] = 0;

}

while (n\_balls--)

{

int t\_row = 0;

int t\_col = 0;

while (arr[t\_row][t\_col] != 0)

{

t\_row = 1 + rand() % row;

t\_col = 1 + rand() % col;

}

arr[t\_row][t\_col] = 1 + rand() % 7;

}

}

int is\_over(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col)

{

for (int i = 1; i <= row; i++)

for (int j = 1; j <= col; j++)

if (arr[i][j] == 0)

return 0;

return 1;

}

int array\_print\_move\_img(int arr[MAX\_ROW + 2][MAX\_COL + 2], int path[MAX\_ROW + 2][MAX\_COL + 2], const int row, const int col, int \*ball\_count, const int can\_keyboard)

{

int row\_from, row\_to, col\_from, col\_to;

int flag = 0;

for (int i = 1; i <= row; i++)

for (int j = 1; j <= col; j++)

path[i][j] = 0;

do

{

if (!read\_position\_mouse\_img(row\_from, col\_from, row, col, can\_keyboard))

return -1;

} while (arr[row\_from][col\_from] <= 0);

showstr(4 \* col\_from - 2, 2 \* row\_from, "◎", 7 + arr[row\_from][col\_from], COLOR\_HWHITE);

do

{

while (1)

{

if (!read\_position\_mouse\_img(row\_to, col\_to, row, col, can\_keyboard))

return -1;

if (arr[row\_to][col\_to] > 0)

{

showstr(4 \* col\_from - 2, 2 \* row\_from, "〇", 7 + arr[row\_from][col\_from], COLOR\_HWHITE);

row\_from = row\_to;

col\_from = col\_to;

showstr(4 \* col\_from - 2, 2 \* row\_from, "◎", 7 + arr[row\_from][col\_from], COLOR\_HWHITE);

continue;

}

break;

}

} while (!path\_find(arr, path, row\_from, col\_from, row\_to, col\_to, flag));

disp\_move\_img(arr, path, row\_from, col\_from, row\_to, col\_to, row\_from, col\_from);

gotoxy(0, 2 + 2 \* row);

array\_move(arr, row\_from, col\_from, row\_to, col\_to);

if (remove(arr, row\_to, col\_to, row, col, ball\_count))

return 2;

return 1;

}

//读入鼠标动作 按下左键返回1 按下右键返回0

int read\_position\_mouse\_img(int &t\_row, int &t\_col, const int row, const int col, const int can\_keyboard = 0)

{

int X = 2, Y = 3;

int ret, maction;

int keycode1, keycode2;

t\_row = 1;

t\_col = 1;

enable\_mouse();

setcolor();

while (1) {

/\* 读鼠标/键盘，返回值为下述操作中的某一种, 当前鼠标位置在<X,Y>处 \*/

ret = read\_keyboard\_and\_mouse(X, Y, maction, keycode1, keycode2);

if (ret == CCT\_MOUSE\_EVENT) {

if (Y > 1 && Y < 2 \* row + 1 && Y % 2 == 0)

t\_row = Y / 2;

if (X < 4 \* col&&X % 4>1)

t\_col = X / 4 + 1;

gotoxy(0, 2 + 2 \* row);

cout << "[当前光标] " << setw(2) << char(t\_row + 'A' - 1) << "行" << setw(2) << t\_col << "列";

switch (maction) {

case MOUSE\_LEFT\_BUTTON\_CLICK: //按下左键

return 1;

case MOUSE\_RIGHT\_BUTTON\_CLICK: //按下右键

return 0;

default:

break;

}

}

else if (can\_keyboard&&ret == CCT\_KEYBOARD\_EVENT) {

switch (keycode1) {

case 27: //ESC

setcursor(CURSOR\_INVISIBLE);

return 0;

case '\r': //回车

setcursor(CURSOR\_INVISIBLE);

return 1;

case 224:

switch (keycode2)

{

case KB\_ARROW\_UP:

if (t\_row > 1)

t\_row--;

break;

case KB\_ARROW\_DOWN:

if (t\_row < row)

t\_row++;

break;

case KB\_ARROW\_RIGHT:

if (t\_col < col)

t\_col++;

break;

case KB\_ARROW\_LEFT:

if (t\_col > 1)

t\_col--;

break;

default:

break;

}

default:

break;

}

gotoxy(0, 2 + 2 \* row);

cout << "[当前光标] " << setw(2) << char(t\_row + 'A' - 1) << "行" << setw(2) << t\_col << "列";

setcursor(CURSOR\_VISIBLE\_FULL);

gotoxy(4 \* t\_col - 2, 2 \* t\_row);

}

}

disable\_mouse(); //禁用鼠标

}

int path\_find(const int arr[MAX\_ROW + 2][MAX\_COL + 2], int path[MAX\_ROW + 2][MAX\_COL + 2], const int row\_from, const int col\_from, const int row\_to, const int col\_to, int &flag\_find)

{

path[row\_from][col\_from] = 1; //先假设走此地

if (row\_from == row\_to && col\_from == col\_to) //到达目的地

flag\_find = 1;

//未到达目的地则继续查找

if (!flag\_find&&arr[row\_from][col\_from + 1] == 0 && path[row\_from][col\_from + 1] == 0) //右

path\_find(arr, path, row\_from, col\_from + 1, row\_to, col\_to, flag\_find);

if (!flag\_find&&arr[row\_from + 1][col\_from] == 0 && path[row\_from + 1][col\_from] == 0) //下

path\_find(arr, path, row\_from + 1, col\_from, row\_to, col\_to, flag\_find);

if (!flag\_find&&arr[row\_from][col\_from - 1] == 0 && path[row\_from][col\_from - 1] == 0) //左

path\_find(arr, path, row\_from, col\_from - 1, row\_to, col\_to, flag\_find);

if (!flag\_find&&arr[row\_from - 1][col\_from] == 0 && path[row\_from - 1][col\_from] == 0) //上

path\_find(arr, path, row\_from - 1, col\_from, row\_to, col\_to, flag\_find);

if (!flag\_find) //此路不通

path[row\_from][col\_from] = 2;

return flag\_find;

}

int remove(int arr[MAX\_ROW + 2][MAX\_COL + 2], const int row\_target, const int col\_target, const int row, const int col, int \*ball\_count)

{

int i, from, to, count = 1, flag = 0;

//纵向

for (from = row\_target - 1; from >= 1 && arr[from][col\_target] == arr[row\_target][col\_target]; from--)

count++;

for (to = row\_target + 1; to <= row && arr[to][col\_target] == arr[row\_target][col\_target]; to++)

count++;

if (count >= 5)

{

flag = 1;

for (i = from + 1; i < to; i++)

{

if (i == row\_target)

continue;

arr[i][col\_target] = 0;

}

if (ball\_count)

ball\_count[arr[row\_target][col\_target]] += count - 1;

}

//横向

count = 1;

for (from = col\_target - 1; from >= 1 && arr[row\_target][from] == arr[row\_target][col\_target]; from--)

count++;

for (to = col\_target + 1; to <= col && arr[row\_target][to] == arr[row\_target][col\_target]; to++)

count++;

if (count >= 5)

{

flag = 1;

for (i = from + 1; i < to; i++)

{

if (i == col\_target)

continue;

arr[row\_target][i] = 0;

}

if (ball\_count)

ball\_count[arr[row\_target][col\_target]] += count - 1;

}

//斜向-左上到右下

count = 1;

for (from = -1; col\_target + from >= 1 && row\_target + from >= 1

&& arr[row\_target + from][col\_target + from] == arr[row\_target][col\_target]; from--)

count++;

for (to = 1; col\_target + to <= col && row\_target + to <= row

&& arr[row\_target + to][col\_target + to] == arr[row\_target][col\_target]; to++)

count++;

if (count >= 5)

{

flag = 1;

for (i = from + 1; i < to; i++)

{

if (i == 0)

continue;

arr[row\_target + i][col\_target + i] = 0;

}

if (ball\_count)

ball\_count[arr[row\_target][col\_target]] += count - 1;

}

//斜向-左下到右上

count = 1;

for (from = -1; col\_target + from >= 1 && row\_target + from <= row

&& arr[row\_target - from][col\_target + from] == arr[row\_target][col\_target]; from--)

count++;

for (to = 1; col\_target + to <= col && row\_target + to >= 1

&& arr[row\_target - to][col\_target + to] == arr[row\_target][col\_target]; to++)

count++;

if (count >= 5)

{

flag = 1;

for (i = from + 1; i < to; i++)

arr[row\_target - i][col\_target + i] = 0;

if (ball\_count)

ball\_count[arr[row\_target][col\_target]] += count - 1;

}

if (flag)

{

ball\_count[arr[row\_target][col\_target]]++;

arr[row\_target][col\_target] = 0;

return 1;

}

return 0;

}