彩球游戏实验报告

班级：软件5班

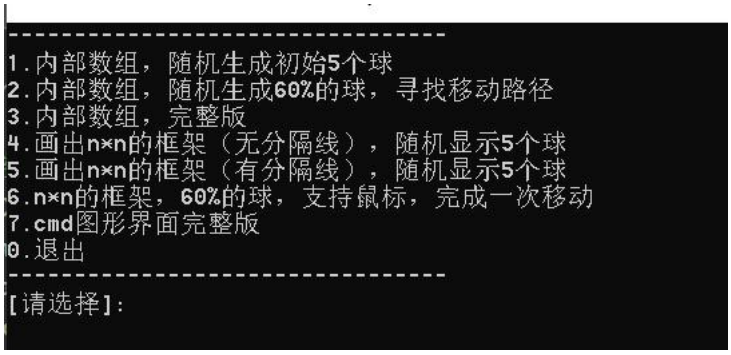
学号：2152402

姓名：段婷婷

完成日期：2022.12.16

1. 题目及基本要求描述

在 cmd 伪图形界面下实现与 Windows 版 Color linez 类似功能的小游戏 ，用菜单方式进行选择，实现以下选项功能：



选项 1：输入行列后，在规定范围内随机生成五个球的位置，然后打印整个内部数组
选项 2：输入行列后，在规定范围内随机生成 60%的球的位置，然后输入要移动球的起始坐标及目的坐标，找出将球移动过去的路径
选项 3：在选项1和2的基础上，用内部数组方式完整地实现彩球游戏
选项 4：在 cmd 伪图形界面上画出框架（无分隔线）及初始的五个球
选项 5：在 cmd 伪图形界面上画出框架（有分隔线）及初始的五个球
选项 6：在 cmd 伪图形界面上显示 60%的球，用鼠标选择要移动的球及目的位置，完成一个移动
选项 7：在 cmd 伪图像界面上完整地实现彩球游戏

2. 整体设计思路

2.1. 各个源文件名以及作用

整个程序由7个源文件组成：cmd_console_tools.h，cmd_console_tools.cpp，90-b2.h，90-b2-main.cpp，90-b2-base.cpp，90-b2-console.cpp，90-b2-tools.cpp。

其中 cmd_console_tools.h 和 cmd_console_tools.cpp分别是给定的伪图形界面下基本功能函数的函数声明和具体实现。

90-b2.h 用于声明90-b2-main.cpp，90-b2-base.cpp，90-b2-console.cpp，90-b2-tools.cpp中需要用到的函数。

90-b2-main.cpp 用于实现菜单的显示、读入正确的选项后选择相应的处理函数。

90-b2-base.cpp 用于实现内部数组方式实现彩球游戏需要用到的函数。

90-b2-console.cpp 用于实现cmd图形界面方式实现彩球游戏需要用到的函数。

90-b2-tools.cpp 用于实现内部数组方式和cmd图形界面方式所用的公共函数。

2.2. 实现内部数组方式的执行函数的设计思路

主要分为以下六类：

1. main函数中根据menu()返回值直接调用的函数。
2. 初始化函数：
   初始化游戏参数的函数。
3. 搜索函数：
   记忆化搜索找到彩球移动路径的函数。
4. 完成移动后的工作的函数：
   移动完成后，判断有无可以消去的彩球、生成新的彩球的函数。
5. 输出函数：
   输出移动完成/新彩球生成后的内部数组的函数。
6. 辅助以上函数工作的函数。

以上函数的具体介绍见下一个部分【3.主要功能的实现】

2.3. 实现cmd图形界面方式的执行函数的设计思路

主要分为以下六类：

1. main函数中根据menu()返回值直接调用的函数。
2. 初始化函数：
   初始化游戏参数的函数。
3. 搜索函数：
   记忆化搜索找到彩球移动路径的函数。
4. 完成移动后的工作的函数：
   移动完成后，判断有无可以消去的彩球、生成新的彩球的函数。
5. 显示函数：
   显示移动过程、图形化界面的函数。
6. 辅助以上函数工作的函数。

以上函数的具体介绍见下一个部分【3.主要功能的实现】

3. 主要功能的实现

3.1. 子题目7的执行函数的功能实现

3.1.1. main函数种根据menu()返回值直接调用的函数：

①函数声明: void complete_solution();
②函数功能：当用户选择菜单项7【cmd图形界面完整版】时由main函数直接调用

3.1.2. 初始化函数

（1）①函数声明：void input_XY(int& X, int& Y);
②函数功能：用于输入游戏的行数与列数
（2）①函数声明：void init_balls(int(*map)[9], const int X, const int Y, const int cnt);
②函数功能：初始化生成cnt个球，并存储在二维数组map中
（3）①函数声明：void init_cnt_color(int(*map)[9], const int X, const int Y, int* cnt_color);
②参数：cnt_color[]: 用于记录各种颜色的彩球数目
③函数功能：根据二维数组map的值来初始化一维数组cnt_color

3.1.3. 搜索函数

①函数声明：bool dfs(const int fr_x, const int fr_y, const int to_x, const int to_y, bool(*vis)[9], const int(*map)[9], const int X, const int Y, const int color, int(*path)[9], const int step, int(*rem)[9]);
②参数：fr_x/fr_y: 当前点坐标
to_x/to_y: 终点坐标
vis[][]: 记录各个点是否在当前路线中
map[][]: 记录地图
X/Y: 地图的行数/列数
color: 待移动彩球的颜色
step: 从起点走了多少步到当前点
rem[][]: 从起点走到某个点所需要的最少步数（不可达为-1） 【记忆化】
path[][]: 从起点走到某个点的最短路径中，该点的上一个点 【路径记录】
③返回值：若可达则返回1，不可达则返回0.
④函数功能：通过记忆化搜索的方法，得到两点之间的**最短路径**.

3.1.4. 完成移动后的工作的函数

（1）①函数声明：bool judge_transverse(int(*map)[9], const int X, const int Y, const int x, const int y);
②函数功能：判断从(x,y)横向往右有无至少5个相同颜色的彩球
（2）①函数声明：bool judge_portrait(int(*map)[9], const int X, const int Y, const int x, const int y);
②函数功能：判断从(x,y)纵向往下有无至少5个相同颜色的彩球
（3）①函数声明：bool judge_diagonal1(int(*map)[9], const int X, const int Y, const int x, const int y);
②函数功能：判断从(x,y)在主对角线往右下方向有无至少6个相同颜色的彩球
（4）①函数声明：bool judge_diagonal2(int(*map)[9], const int X, const int Y, const int x, const int y);
②函数功能：判断从(x,y)在副对角线往坐下方向有无至少6个相同颜色的彩球
（5）①函数声明：int judge(int(*map)[9], const int X, const int Y, int* cnt_color, bool show = 0);
②函数功能：判断游戏地图中有无可以消去的彩球，若有，则消去并返回消去的彩球颜色；否则返回0.
（6）①函数声明：void new_balls(int(*map)[9], const int X, const int Y, const int cnt, const int* next_color, int* cnt_color, bool show);

②函数功能：在彩球移动后，按照next_color[]的颜色生成新的球。

3.1.5.显示函数

（1）①函数声明：void show_graph(const int(*map)[9], const int X, const int Y, const int COL, const int ROW);

②函数功能：显示彩球游戏的游戏地图

（2）①函数声明：void show_points(const int x, const int y, const int points);

②函数功能：显示游戏分数

（3）①函数声明：void show_next_color(const int x, const int y, const int* next_color);

②函数功能：预告下一次生成的新彩球的颜色

（4）①函数声明：void show_proportion(const int x, const int y, const int* cnt_color, const int cnt_tot, const int* cnt_delete);

②函数功能：显示当前地图中，各种不同颜色的彩球的数目和占比以及已经消去的数目。

（5）①函数声明： void move_by_path(const int fr_x, const int fr_y, const int to_x, const int to_y, bool(*vis)[9], int(*map)[9], const int X, const int Y, int(*path)[9]);

②函数功能：根据path[][]记录的路径，以图形方式输出彩球的移动过程。

3.1.6. 辅助函数

（1）①函数声明：void wait_for_end();

②函数功能：当菜单项功能完成后，输出提示语句，完成输入End就结束当前菜单项的功能。

（2）①函数声明：void wait_for_enter(const char* s);

②函数功能：实现按回车键继续的功能。

3.2 子题目7的实现：

①首先定义选项7需要用到的变量并且调用相应的函数进行初始化，显示初始化的图像。

②显示分数、预告彩球坐标、显示彩球比例。

③然后持续读取光标移动与点击，得到要移动的彩球的起始和目标坐标。

④接着调用搜索函数来找到路径：若可达，则以图形化的方式输出移动过程；否则输出提示语句，并返回步骤②。

⑤移动完成后，调用相应函数消去可以消去的彩球并累计分数，按照预告的颜色生成新的彩球。若游戏仍然可以继续，则返回②；若游戏结束，则执行⑥。

⑥输出游戏结束的原因，并且调用wait_for_end()函数，结束菜单项。

4. 调试过程碰到的问题

4.1.

(1)**问题**：发现在游戏中，只有第一次移动能正确地找到路径，后续移动出错。
(2)**差错方法与错处**：检查在每一次调用搜索函数中，其用到的变量，如path[][]、rem[][]等，发现对这些变量的初始化放在循环外。换言之，这些变量的初始化在整个程序中只做了一次，故只有第一次移动正确。
(3)**解决方法**：每次调用搜索函数前，都需要对其用到的变量进行初始化。
(4)**小结**：在重复调用一个函数时，注意变量是否及时初始化了。

4.2.

(1)**问题**：在找到路径的基础上，如何搜索得到最短路径？
(2)**解决方法**：在参数中多加上step与rem[][]两个，分别记录从起点到当前点的步数，以及从起点到各个点的最短路径长度。采用**记忆化搜索**的方式，这样既能找到最短路径，也不会重复搜索，事半功倍。
(3)**小结**：在目前会的方法不能满足功能需求时，可以考虑改进或者学习新的方法。

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

在写代码的过程中最最重要的是理清逻辑思路，但是也不必要求自己在一开始就考虑好、统筹全所有的情况，在过程中敢于推翻自己、调整思路、从新架构也是非常重要的。

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？

在完成两个综合题的过程中，我考虑了前后小题的关联关系，并且尽可能做到后面小题有效利用前面小题已完成的代码。

如何更好地重用代码？

①在写代码时就要对代码有清晰的划分，明确每个部分实现的功能，方便后面需要相似的功能时可以快速地找到之前写过的部分，重用代码。

②将一部分实现某一特定功能的代码写成函数有利于重用代码。

③对于函数，尽量做到函数中用到的变量都是参数表中的或函数内定义的，尽量少用全局变量，这样重用代码时只需要移植函数部分，在其内部进行需要的改动即可。

④对于同样意义的变量统一命名，并且命名要能清楚地读出其含义，这样可以在重用时减少改动的部分，便于重用代码。

6. 附件：源程序

(1)90-b2-console.cpp

```
/*2152402 软件 段婷婷*/
#include<iostream>
#include<iomanip>
#include<windows.h>
#include<fstream>
#include "cmd_console_tools.h"
#include "90-b2.h"

using namespace std;
const int TYPE = 7;

void show_graph1(const int(*map)[9], const int X,
const int Y)
{
    cct_cls();
    const int COL = 50, ROW = 20;
    cct_setconsoleborder(COL, ROW, COL, ROW);
    cct_gotoxy(0, 0);
    cout << "屏幕：" << ROW << "行" << COL << "列";
    cct_setfontsize("新宋体", 40, 20);
    int x = 1, y = 0;
    cct_showstr(y, x, "╔", 7, 0, 1, 2);
    for (int j = 0;j <= Y;++j) {
        y += 2;
        cct_showstr(y, x, "═", 7, 0, 1, 3);
    }
    cct_showstr(y, x, "╗", 7, 0, 1, 3);

    for (int i = 0;i < X;++i) {
        ++x;
        y = 0;
        cct_showstr(y, x, "║", 7, 0, 1, 3);
        for (int j = 0;j < Y;++j) {
            y += 2;
            if (map[i][j] == 0)
                cct_showstr(y, x, " ", 7, 0, 1, 3);
            else
                cct_showstr(y, x, "○", map[i][j] + 7,
7, 1);
        }
        y += 2;
        cct_showstr(y, x, "║", 7, 0, 1, 3);
    }
    ++x;
    y = 0;
    cct_showstr(y, x, "╚", 7, 0, 1, 3);
    for (int j = 0;j <= Y;++j) {
        y += 2;
        cct_showstr(y, x, "═", 7, 0, 1, 3);
    }
    cct_showstr(y, x, "╝", 7, 0, 1, 3);
}
void show_graph(const int(*map)[9], const int X, const
int Y, const int COL, const int ROW)
{
    cct_cls();
    cct_setconsoleborder(COL, ROW, COL, ROW + 1000);
    cct_gotoxy(0, 0);
    cout << "屏幕：" << ROW << "行" << COL << "列";
    cct_setfontsize("新宋体", 26, 13);
    int x = 1, y = 0;
    cct_showstr(y, x, "╔", 7, 0, 1);
    y += 2;
    cct_showstr(y, x, "═╦", 7, 0, Y - 1);
    y += 4 * (Y - 1);
    cct_showstr(y, x, "═╗", 7, 0, 1);
    for (int i = 0;i < X - 1;++i) {
        ++x;
        y = 0;
        cct_showstr(y, x, "║  ", 7, 0, Y);
        y += (Y) * 4 - 1;
        cct_showstr(y, x, " ║", 7, 0, 1);
        ++x;
        y = 0;
        cct_showstr(y, x, "╠", 7, 0, 1);
        y += 2;
        cct_showstr(y, x, "═╬", 7, 0, Y - 1);
        y += (Y - 1) * 4;
        cct_showstr(y, x, "═╣", 7, 0, 1);
    }
    ++x;
    y = 0;
    cct_showstr(y, x, "║  ", 7, 0, Y);
    y += (Y) * 4 - 1;
    cct_showstr(y, x, " ║", 7, 0, 1);
    ++x;
    y = 0;
    cct_showstr(y, x, "╚", 7, 0, 1);
    y += 2;
    cct_showstr(y, x, "═╩", 7, 0, Y - 1);
    y += 4 * (Y - 1);
    cct_showstr(y, x, "═╝", 7, 0, 1);
    for (int i = 0;i < X;++i)
        for (int j = 0;j < Y;++j)
            if (map[i][j] != 0) {
                cct_showstr(2 + 4 * j, 2 + 2 * i, "○",
7 + map[i][j], 7);
            }
    cct_gotoxy(0, 2 * X + 1);
}
void draw_for_opt4()
{
    int map[9][9], X, Y;
    memset(map, 0, sizeof(map));
    input_XY(X, Y);
    init_balls(map, X, Y, 5);
    output_map(map, X, Y, 0);

    wait_for_enter("按回车键显示图形...");
    show_graph1(map, X, Y);
    wait_for_end();
}

void draw_for_opt5()
{
    int map[9][9], X, Y;
    memset(map, 0, sizeof(map));
    input_XY(X, Y);
    init_balls(map, X, Y, 5);
    output_map(map, X, Y, 0);

    wait_for_enter("按回车键显示图形...");
    show_graph(map, X, Y, 50, 25);
    wait_for_end();
}

void draw_for_opt6()
{
    int map[9][9], X, Y, path[9][9], rem[9][9];
    bool vis[9][9];
    memset(map, 0, sizeof(map));

    input_XY(X, Y);
```

```
    init_balls(map, X, Y, int(X * Y * 0.6));
    show_graph(map, X, Y, 50, 25);
    cct_enable_mouse();
    cct_setcursor(CURSOR_INVISIBLE);
    int X1 = 0, Y1 = 0, ret, maction, keycode1,
keycode2;
    int fr_x = -1, fr_y = -1, to_x, to_y;
    bool tag = 0;
    while (1) {
        while (1) {
            ret = cct_read_keyboard_and_mouse(X1, Y1,
maction, keycode1, keycode2);
            if (ret == CCT_MOUSE_EVENT) {
                cct_gotoxy(0, X * 2 + 1);
                int x = (Y1 - 2) / 2, y = (X1 - 2) /
4;
                cct_setcolor();
                if (x < X && y < Y)
                    cout << endl << "[当前光标] " <<
char('A' + x) << "行" << char('1' + y) << "列";
                if (maction ==
MOUSE_LEFT_BUTTON_CLICK && (map[x][y] || tag)) {
                    if (map[x][y] != 0) {
                        if (fr_x != -1) {
                            cct_showstr(2 + 4 * fr_y,
2 + 2 * fr_x, "○", 7 + map[fr_x][fr_y], 7);
                        }
                        fr_x = x;
                        fr_y = y;
                        cct_showstr(2 + 4 * fr_y, 2 +
2 * fr_x, "◎", 7 + map[fr_x][fr_y], 7);
                        tag = 1;
                    }
                    if (tag && map[x][y] == 0) {
                        to_x = x;
                        to_y = y;
                        break;
                    }
                }
            }
        }
        memset(vis, 0, sizeof(vis));
        memset(path, -1, sizeof(path));
        memset(rem, -1, sizeof(rem));
        if (dfs(fr_x, fr_y, to_x, to_y, vis, map, X,
Y, map[fr_x][fr_y], path, 0, rem)) {
            move_by_path(fr_x, fr_y, to_x, to_y, vis,
map, X, Y, path);
            cct_gotoxy(0, X * 2 + 2);
            cct_setcolor();

            break;
        }
        else {
            cct_setcolor();
            cct_gotoxy(0, X * 2 + 3);
            cout << "[错误] 无法从[" << char('A' +
fr_x) << char('1' + fr_y) << "]移到[" << char('A' + to_x)
<< char('1' + to_y) << "]";
            Sleep(300);
            cct_gotoxy(0, X * 2 + 3);
            cout << setw(30) << " ";
        }
    }
    cct_gotoxy(0, X * 2 + 1);
    cct_setcolor();

    wait_for_end();
}
void move_by_path(const int fr_x, const int fr_y,
const int to_x, const int to_y, bool(*vis)[9],
int(*map)[9], const int X, const int Y, int(*path)[9])
{
    map[to_x][to_y] = map[fr_x][fr_y];
    map[fr_x][fr_y] = 0;
    int cur_x = fr_x, cur_y = fr_y;
    int last_x = -1, last_y = -1;
    while (1) {
        int d_x = 0, d_y = 0;
        if (last_x != -1 && last_y != -1) {
            d_x = last_x - cur_x;
            d_y = last_y - cur_y;
        }
        cct_showstr(2 + 4 * cur_y + d_y * 2, 2 + 2 *
cur_x + d_x, "◎", 7 + map[to_x][to_y], 7);
        Sleep(10);
        if (d_y != 0)
            cct_showstr(2 + 4 * cur_y + d_y * 2, 2 +
2 * cur_x + d_x, "┃", 7, 0);
        else if (d_x != 0)
            cct_showstr(2 + 4 * cur_y + d_y, 2 + 2 *
cur_x + d_x, "═", 7, 0);

        cct_showstr(2 + 4 * cur_y, 2 + 2 * cur_x, "◎",
7 + map[to_x][to_y], 7);
        Sleep(10);
        if (cur_x == to_x && cur_y == to_y)
            break;
        cct_showstr(2 + 4 * cur_y, 2 + 2 * cur_x, "  ",
7, 7);
        last_x = cur_x;
        last_y = cur_y;
        int tmp = path[cur_x][cur_y];
        if (tmp < 0) break;
        cur_y = tmp % Y;
        cur_x = (tmp - cur_y) / Y;
    }
    cct_showstr(2 + 4 * to_y, 2 + 2 * to_x, "○", 7 +
map[to_x][to_y], 7);

    cct_gotoxy(0, X * 2 + 2);
    cct_setcolor();
}
void show_points(const int x, const int y, const int
points)
{
    cct_showstr(y, x, "┏━━━━━━━━┓", 7, 0);
    cct_showstr(y, x + 1, "┃得分:    ┃", 7, 0);
    cct_showstr(y, x + 2, "┗━━━━━━━━┛", 7, 0);
    cct_showint(y + 9, x + 1, points, 7, 0);
}
void show_next_color(const int x, const int y, const
int* next_color)
{
    cct_showstr(y, x, "┏━┳━┳━┳━┓", 7, 0);
    cct_showstr(y, x + 1, "┃  ┃  ┃  ┃  ┃", 7, 0);
    cct_showstr(y, x + 2, "┗━┻━┻━┻━┛", 7, 0);
    for (int i = 0;i < 3;++i){
        int col = next_color[i];
        cct_showstr(y + 2 + 4 * i, x + 1, "○", 7 + col,
7);
    }
}
void show_proportion(const int x, const int y, const
int* cnt_color, const int cnt_tot, const int*
cnt_delete)
{
    cct_showstr(y, x, "┏━━━━━━━━━━━━━━━━━━━━┓",
7, 0);
    for (int i = 0;i <= TYPE;++i) {
        cct_showstr(y, x + i + 1, "┃
┃", 7, 0);
```

```
        cct_showstr(y + 2, x + i + 1, "○", 7 + i, 7);
        cct_showstr(y + 4, x + i + 1, ":", 7, 0);
        int cnt = i == 0 ? (cnt_tot - cnt_color[0]) :
cnt_color[i];
        if (cnt < 10) {
            cct_showstr(y + 5, x + i + 1, "0", 7, 0);
            cct_showint(y + 6, x + i + 1, cnt, 7, 0);
        }
        else
            cct_showint(y + 5, x + i + 1, cnt, 7, 0);
        double pro = 1.0 * cnt / cnt_tot;
        char pro_s[15];
        pro_s[0] = '/';
        pro_s[1] = '(';
        if (int(pro * 10) % 10 == 0)
            pro_s[2] = '0';
        else pro_s[2] = '0' + (int(pro * 10) % 10);
        pro_s[3] = '0' + (int(pro * 100) % 10);
        pro_s[4] = '.';
        pro_s[5] = '0' + (int(pro * 1000) % 10);
        pro_s[6] = '0' + (int(pro * 10000) % 10);
        pro_s[7] = '%';
        pro_s[8] = ')';
        pro_s[9] = '\0';
        cct_showstr(y + 7, x + i + 1, pro_s, 7, 0);
        cct_showstr(y + 15, x + i + 1, ") 消除-", 7,
0);
        cct_showint(y + 22, x + i + 1, cnt_delete[i],
7, 0);

    }
    cct_showstr(y, x + 9,
" ╚═══════════════════════╝ ", 7, 0);

}
void complete_solution()
{
    int map[9][9], X, Y;
    int next_color[3], cnt_color[TYPE + 1],
cnt_delete[TYPE + 1];
    int points = 0;
    int rem[9][9], path[9][9];
    bool vis[9][9];
    memset(map, 0, sizeof(map));

    input_XY(X, Y);
    init_balls(map, X, Y, 5);
    show_graph(map, X, Y, 70, 25);
    init_cnt_color(map, X, Y, cnt_color);
    cct_enable_mouse();
    cct_setcursor(CURSOR_INVISIBLE);
    for (int i = 0;i < 3;++i) {
        next_color[i] = rand() % TYPE + 1;
        //cout << next_color[i] << " ";
    }
    for (int i = 0;i <= TYPE;++i)
        cnt_delete[i] = 0;

    int X1 = 0, Y1 = 0, ret, maction, keycode1,
keycode2;
    int fr_x = -1, fr_y = -1, to_x, to_y;
    bool tag = 0;

    while (cnt_color[0] < X * Y && cnt_color[0] > 0)
{
        show_points(1, 4 * Y + 4, points);
        show_next_color(5, 4 * Y + 4, next_color);
        show_proportion(9, 4 * Y + 4, cnt_color, X *
Y, cnt_delete);

        while (1) {
            ret = cct_read_keyboard_and_mouse(X1, Y1,
maction, keycode1, keycode2);
            if (ret == CCT_MOUSE_EVENT) {
                cct_gotoxy(0, X * 2 + 1);
                int x = (Y1 - 2) / 2, y = (X1 - 2) /
4;
                cct_setcolor();
                if (x < X && y < Y)
                    cout << endl << "[当前光标] " <<
char('A' + x) << "行" << char('1' + y) << "列";
                if (maction ==
MOUSE_LEFT_BUTTON_CLICK && (map[x][y] || tag)) {
                    if (map[x][y] != 0) {
                        if (fr_x != -1) {
                            cct_showstr(2 + 4 * fr_y,
2 + 2 * fr_x, "○", 7 + map[fr_x][fr_y], 7);
                        }
                        fr_x = x;
                        fr_y = y;
                        cct_showstr(2 + 4 * fr_y, 2 +
2 * fr_x, "◎", 7 + map[fr_x][fr_y], 7);
                        tag = 1;
                    }
                    if (tag && map[x][y] == 0) {
                        to_x = x;
                        to_y = y;
                        break;
                    }
                }
            }
        }

        memset(vis, 0, sizeof(vis));
        memset(path, -1, sizeof(path));
        memset(rem, -1, sizeof(rem));

        if (dfs(fr_x, fr_y, to_x, to_y, vis, map, X,
Y, map[fr_x][fr_y], path, 0, rem)) {
            move_by_path(fr_x, fr_y, to_x, to_y, vis,
map, X, Y, path);
            //break;

            int col = judge(map, X, Y, cnt_color, 1);

            if (col) {
                cnt_delete[col] += 5;
                points += 12;
            }
            else {
                new_balls(map, X, Y, 3, next_color,
cnt_color, 1);
                col = judge(map, X, Y, cnt_color, 1);
                if (col) {
                    cnt_delete[col] += 5;
                    points += 12;
                }
            }
            for (int i = 0;i < 3;++i) {
                next_color[i] = rand() % TYPE + 1;
            }
        }
        else {
            cct_setcolor();
            cct_gotoxy(0, X * 2 + 3);
            cout << "[错误] 无法从[" << char('A' +
fr_x) <<char('1' + fr_y) <<"]移到[" <<char('A' + to_x)
<< char('1' + to_y) << "]";
            Sleep(300);
            cct_gotoxy(0, X * 2 + 3);
            cout << setw(30) << " ";
        }
```

```
    }
    show_points(1, 4 * Y + 4, points);
    show_next_color(5, 4 * Y + 4, next_color);
    show_proportion(9, 4 * Y + 4, cnt_color, X * Y,
cnt_delete);

    cct_gotoxy(0, X * 2 + 4);
    cct_setcolor();

    if (cnt_color[0] != 0)
        cout << "无空位可移，游戏结束！" << endl;
    else
        cout << "无球可移，游戏结束！" << endl;

    wait_for_end();
}
```

(2)90-b2-tools.cpp

```
/*2152402 软件 段婷婷*/
#include<iostream>
#include<iomanip>
#include<windows.h>
#include "cmd_console_tools.h"
using namespace std;
void input_XY(int& X, int& Y) //输入行数和列数
{
    cct_cls();
    int tmp[2];
    const char indi[2][20] = { "请输入行数(7-9)" ,"
请输入列数(7-9)" };
    for (int i = 0;i < 2;++i) {
        while (1) {
            cout << indi[i] << endl;
            cin >> tmp[i];
            if (!cin.good())
            {
                cin.clear();
                cin.ignore(INT_MAX, '\n');
            }
            if (tmp[i] >= 7 && tmp[i] <= 9)
                break;
        }
    }
    X = tmp[0];
    Y = tmp[1];
}
void init_balls(int(*map)[9], const int X, const int
Y, const int cnt) //初始生成 cnt 个球
{
    int x, y, cur_cnt = 0;
    while (cur_cnt < cnt)
        while (1) {
            x = rand() % X;
            y = rand() % Y;
            if (map[x][y] == 0) {
                map[x][y] = rand() % 7 + 1;
                ++cur_cnt;
                break;
            }
        }
}
void new_balls(int(*map)[9], const int X, const int Y,
const int cnt, const int* next_color, int* cnt_color,
bool show)
{
    int n = min(cnt, X * Y - cnt_color[0]);
    for (int i = 0;i < n;++i) {
        while (1) {
            int x = rand() % X, y = rand() % Y;
            if (map[x][y] == 0) {
                map[x][y] = next_color[i];
                if (show)
                    cct_showstr(2 + 4 * y, 2 + 2 * x,
"○", 7 + map[x][y], 7);
                ++cnt_color[next_color[i]];
                break;
            }
        }
    }
    cnt_color[0] += n;
}
void wait_for_end() //输入 End 结束
{
    cct_setcolor();
    cout << endl << "本小题结束，请输入 End 继续....";
    int x, y;
    cct_getxy(x, y);
    char s[10];
    const char end[4] = "END";
    fgets(s, 9, stdin);
    while (1) {
        cct_gotoxy(x, y);
        cout << setw(20) << " ";
        cct_gotoxy(x, y);
        fgets(s, 9, stdin);
        bool tag = 1;
        for (int i = 0;i < 3;++i)
            if (s[i] != end[i] && s[i] != end[i] + 32)
{
                tag = 0;
                break;
            }
        if (tag)
            break;
        cct_gotoxy(0, y + 1);
        cout << "输入错误，请重新输入";
        Sleep(100);
        cct_gotoxy(0, y + 1);
        cout << setw(20) << " ";
    }
}
void chlwr(char& ch)
{
    if (ch < 'a')
        ch += 32;
}
void wait_for_enter(const char* s)
{
    cout << endl << s;
    int X = 0, Y = 0, ret, maction, keycode1, keycode2;
    while (1)
    {
        ret = cct_read_keyboard_and_mouse(X, Y,
maction, keycode1, keycode2);
        if (ret == CCT_KEYBOARD_EVENT && keycode1 ==
13)
            break;
    }
}

const int DX[4] = { 0,0,1,-1 }, DY[4] = { 1,-1,0,0 };
bool dfs(const int fr_x, const int fr_y, const int to_x,
const int to_y, bool(*vis)[9], const int(*map)[9],
const int X, const int Y, const int color,
int(*path)[9], const int step, int(*rem)[9])
{
    rem[fr_x][fr_y] = step;
    if (fr_x == to_x && fr_y == to_y) {
```

```
        return 1;
    }
    bool ret = 0;
    for (int i = 0;i < 4;++i) {
        int tx = fr_x + DX[i], ty = fr_y + DY[i];
        if (tx < 0 || tx >= X || ty < 0 || ty >= Y ||
map[tx][ty] != 0 || vis[tx][ty])
            continue;
        vis[tx][ty] = 1;
        if ((rem[tx][ty] == -1 || rem[tx][ty] > step
+ 1) && dfs(tx, ty, to_x, to_y, vis, map, X, Y, color,
path, step + 1, rem)) {
            path[fr_x][fr_y] = tx * Y + ty;
            ret = 1;
            //return 1;
        }
        vis[tx][ty] = 0;
    }
    return ret;
}
bool judge_transverse(int(*map)[9], const int X,
const int Y, const int x, const int y)
{
    for (int j = y;j < y + 5;++j)
        if (map[x][j] != map[x][y])
            return 0;
    return 1;
}
bool judge_portrait(int(*map)[9], const int X, const
int Y, const int x, const int y)
{
    for (int i = x;i < x + 5;++i)
        if (map[i][y] != map[x][y])
            return 0;
    return 1;
}
bool judge_diagonal1(int(*map)[9], const int X, const
int Y, const int x, const int y)
{
    for (int k = 1;k < 5;++k)
        if (map[x + k][y + k] != map[x][y])
            return 0;
    return 1;
}
bool judge_diagonal2(int(*map)[9], const int X, const
int Y, const int x, const int y)
{
    for (int k = 1;k < 5;++k)
        if (map[x - k][y + k] != map[x][y])
            return 0;
    return 1;
}
int judge(int(*map)[9], const int X, const int Y, int*
cnt_color, bool show)
{
    for (int i = 0;i < X;++i)
        for (int j = 0;j < Y;++j) {
            if (map[i][j] == 0 || cnt_color[map[i][j]]
< 5)
                continue;
            if (j + 4 < Y && judge_transverse(map, X,
Y, i, j)) {
                cnt_color[map[i][j]] -= 5;
                cnt_color[0] -= 5;
                int ret = map[i][j];
                for (int y = j;y < j + 5;++y) {
                    map[i][y] = 0;
                    if (show)
                        cct_showstr(2 + 4 * y, 2 + 2
* i, "  ", 7, 7);
                }
                return ret;
            }
            else if (i + 4 < X && judge_portrait(map,
X, Y, i, j)) {
                cnt_color[map[i][j]] -= 5;
                cnt_color[0] -= 5;
                int ret = map[i][j];
                for (int x = i;x < i + 5;++x) {
                    map[x][j] = 0;
                    if (show)
                        cct_showstr(2 + 4 * j, 2 + 2
* x, "  ", 7, 7);
                }
                return ret;
            }
            else if (i + 4 < X && j + 4 < Y &&
judge_diagonal1(map, X, Y, i, j)) {
                cnt_color[map[i][j]] -= 5;
                cnt_color[0] -= 5;
                int ret = map[i][j];
                for (int k = 0;k < 5;++k) {
                    map[i + k][j + k] = 0;
                    if (show)
                        cct_showstr(2 + 4 * (j + k),
2 + 2 * (i + k), "  ", 7, 7);
                }
                return ret;
            }
            else if (i - 4 >= 0 && j + 4 < Y &&
judge_diagonal2(map, X, Y, i, j)) {
                cnt_color[map[i][j]] -= 5;
                cnt_color[0] -= 5;
                int ret = map[i][j];
                for (int k = 0;k < 5;++k) {
                    map[i - k][j + k] = 0;
                    if (show)
                        cct_showstr(2 + 4 * (j + k),
2 + 2 * (i - k), "  ", 7, 7);
                }
                return ret;
            }
        }
    return 0;
}

const int TYPE = 7;
void init_cnt_color(int(*map)[9], const int X, const
int Y, int* cnt_color)
{
    for (int i = 0;i <= TYPE;++i)
        cnt_color[i] = 0;
    for (int i = 0;i < X;++i)
        for (int j = 0;j < Y;++j)
            if (map[i][j] != 0)
                ++cnt_color[map[i][j]];
    for (int i = 1;i <= TYPE;++i)
        cnt_color[0] += cnt_color[i];
}
```