同济大学

《高级语言程序设计》

实验报告

报告名称：彩球游戏（MagicBall）

班　　级：信05

学　　号：2353814

姓　　名：马小龙

日　　期：2024年6月15日

1. 题目

1.1.题目描述

用伪图形界面方式实现彩球游戏

游戏规则：

1.游戏区域为5*5~9*9，共有9种颜色的彩球随机出现，初始随机填满，随机概率相同

2. 消除规则为横向/纵向连续颜色超过3个，每消除1个球计1分，如果横纵向同时存在，则分别计算是否超过三个

3. 消除后，空位上方的球垂直方向落下，填补空位，最上方的空位再用随机颜色的彩球填满

4. 初始填满后，先判断是否有立即可消除的项，如果有，则立即消除/填充/再消除/再填充…，直到无可消除项才停止，停止前的所有消除项不计分

5. 无可消除项后，遍历整个游戏区域，将可互换的球用绿圈标识出来

6. 用鼠标选择可互换的球（再按一次则取消选择），再选择邻近的另一个可互换的球，就能进行交换；对象交换后进行消除/填充…，直到无消除项为止

程序需要实现以下10个功能：

1.内部数组，生成初始状态，寻找是否有初始可消除项

2.内部数组，消除初始可消除项后非0项下落并用0填充

3.内部数组，消除初始可消除项后查找消除提示

4.n*n的框架(无分隔线)，显示初始状态

5.n*n的框架(有分隔线)，显示初始状态

6.n*n的框架(无分隔线)，显示初始状态及初始可消除项

7.n*n的框架(有分隔线)，消除初始可消除项后显示消除提示

8.cmd图形界面完整版(有分隔线，鼠标移动时显示坐标，右键退出)

9.cmd图形界面完整版

0.退出

1.2.题目要求

1、用伪图形界面方式完成彩球游戏(MagicBall)

2、提供90-b2-demo.exe供参考

a) 需设置为旧版控制台，去除快速编辑和插入模式

b) Windows 版的游戏区域是 5*5 9*9 的正方形，自制版是5~9之间任意行列

c) 自制版的可消除球为实心，可互换球为空心双圈，普通球为空心单圈，颜色通过背景色区分

3、附件提供了一个图形界面版的MagicBall 游戏供参考，如果参考游戏的规则和本作业的具钵要求不同，以作业要求为准

4、伪图形界面工具函数集的学习：参见汉诺塔/test-cct

2. 整体设计思路

彩球游戏实现，首先是（5~9）*（5~9）游戏区域中彩球的表现方式，选择用9*9的二维数组表示，通过输入的行数和列数来为二维数组部分元素随机赋值，达到与彩球相类似的结果。

接着为画框和画彩球，借助cct_showstr（）函数，可以实现这一过程，需要注意的是，彩球游戏涉及的边框和彩球均为两字符。画彩球时，需要根据不同情况来判断彩球是●、◎还是○，具体分为两个判断其一是可消除的彩球，通过对数组行列的分析是否连续三个及以上数字相同可以判断，其二为可交换的彩球，通过列举不同情况来进行判断。

最后便是使用鼠标，根据鼠标所处的位置，来计算出对应的是数组哪一个数，进而判断该数是否可以选中，在讨论第二个选中的数是否冲突，最后实现交换这一过程。最后，根据是否有可消除、可交换的彩球或者是否输入右键，来判断是重复执行还是退出执行。

3. 主要功能的实现

3.1.重要功能的实现过程

3.1.1.彩球是否可消除判断

将彩球类比为二维数组后，对二维数组的行、列分别进行判断，来验证是否连续三个及以上数字相同，若有，择将其标记出来。

3.1.2.彩球是否可交换判断

在消除完可消除彩球后，对余下彩球进行两次判断：

其一，若某彩球同行（列）相邻两个彩球相等，寻找其同列（行）相邻彩球是否有相等的，若有，则标记该彩球和所寻得彩球；

其二，若某彩球某一侧有两个连续彩球相等，则寻找其另外三侧彩球是否有相等的，若有，则标记该彩球和所寻得彩球。通过两次判断，便可选找出所有可交换项。可以将数组添加四行四列以方便判断。

3.1.3.彩球消除的实现

在标记完可交换项后，可以将可消除想项对应的数组元素归零，同时，若数组元素归零，cct_showstr（）函数在对应位置打印□，在用cct_showint（），对应位置用设定颜色打印两次空格，实现彩球消除效果。需要注意添加延时，以达到动画效果。

3.1.4.彩球下落的实现

彩球下落彩用的是冒泡法，将同一列归零的元素传递上去，交换位置的同时对对应位置的将原有彩球用cct_showint（）打印空格代替，再在原来空处打印彩球，添加延时，以达到彩球下落的效果。

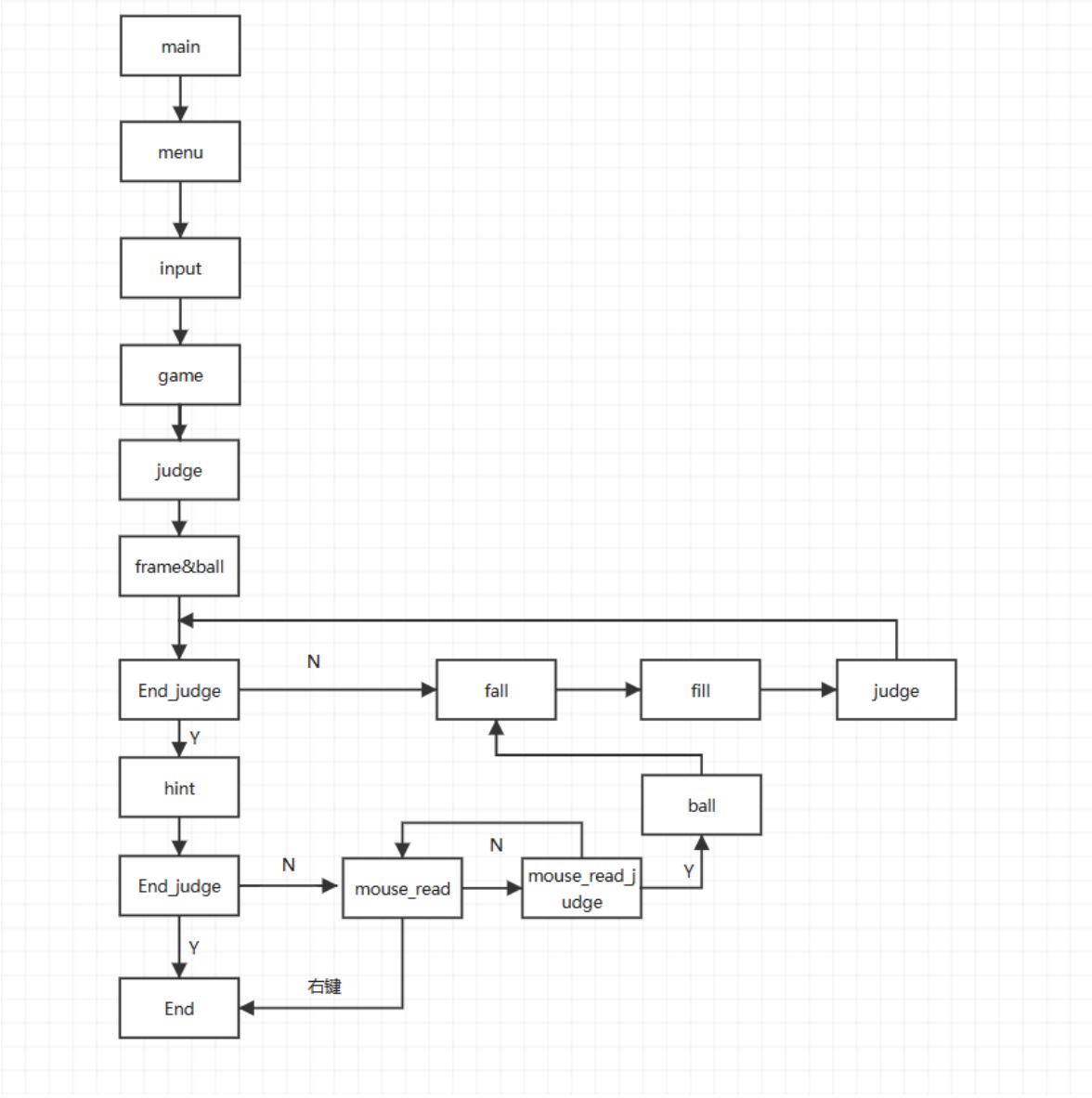
3.1.5.彩球填充的实现

彩球填充时，先对原有数组为零的元素重新随机赋值，再在对应位置用对应颜色打印彩球，添加延时，达到彩球填充的效果。

3.1.6.彩球选取的实现

彩球选取首先需要允许使用鼠标，再借助cct_read_keyboard_and_mouse()函数，读取鼠标所在屏幕的行列，通过计算，得到其在二维数组所指的行和列，通过处理打印来。然后，根据参数&MAction，判断其读取的时鼠标的什么键，若为右键，则退出，若为左键，则继续读取；两次读取后需要进行判断，以保证交换后产生可消除。

3.2.流程框图



4. 调试过程碰到的问题

4.1.数组相关问题

在开始写程序时，我直接使用正常二维数组，但很快发现一个难以解决的地方，这个数组需要在多个函数内使用，且部分函数需要改变其中部分元素，如果直接使用二维数组，虽然做实参、形参时传递很方便，但是，某一函数对它的改变无法影响调用该函数的函数。

我曾想过返回二维数组，但发现以我的知识储备以及所查找的资料来看，这一方法行不通。

后来，我想到用二维指针数组，即定义的二维数组每一个元素均为指针变量，分别指向对应的二维数组元素，传递时传递二维数组指针变量，这样不需要返回，就能改变原函数中二维数组。

4.2.可消除、可交换确定相关问题

可消除、可交换函数的判断、确定不是难题，关键是实现过程，开始时，我的想法是一判断，一输出，即根据判断的结果，即进行输出；但是这样的方式不仅会使单个函数的内容变得非常之多，远远大于50行，而且实现起来相当麻烦。

为了解决这一问题，我添加了“保存”这一步骤，通过定义一个新的二维数组，并用0初始化，将原数组中可消除、可交换的元素，赋值给新的数组的对应元素，这样就可以做到保存，在之后的输出中，可以根据对应元素是否相等来决定输出的内容，每次输出完毕后将新数组全部元素置零，防止影响后续输出。方便起见，我依旧采用二维指针数组的方式，这样就可以将功能不同的函数分离，方便调用。

4.3.坐标问题

使用cct_read_keyboard_and_mouse()函数后，可以读取当前鼠标所在位置的坐标，为计算出相对应的行和列，我将画图时所用的坐标公式进行逆运算，计算出double型数据：dI = (Y - 2.0) / (interval / 2)、 dJ = (X - 2.0) / interval（interval为4）。再将其强转为int型数据I = int(dI)、J = int(dJ)，算出对应行和列，通过判断int型和double是否相等，来确定显示坐标还是“位置非法”。

但很快发现一个问题，即选取彩球的一半时显示坐标，另一半是现实“位置非法”。这时因为一个彩球占了两个字符，因而会出错，因此我将判断条件改为(I == dI || I == dI - 0.5) && (J == dJ || J == dJ - 0.25)，这样就能正确显示。

4.4.鼠标选取后（非右键）的判断问题

鼠标选取后会出现三种情况：选取的相同，选取的交换后不会出现可消除项，选取的非相邻。

第一种情况解决方法较为简单，只需比较行列坐标是否相同。

第二种情况可以重新定义两个二维数组，将原数组的值赋给其中一个新数组，另一个用0初始化。再将被赋值的数组对应元素交换，后通过judge函数对两个新数组判断，判断后，若另一个新数组元素不全为零，则表明选取正确，反之选取错误。

第三种情况，开始时我是根据行±1相等且列±1相等来判断选取的两个值合理，但很快发现这种判断不能避免选取的是四角的情况，于是，我换用行之差的平方与列之差的平方之和是否为大于1来判断，

不大于1则进行下一步，大于1则将第二个点的替换为第一个点，重新第二个点。

4.5.游戏结束语处理问题

游戏结束需要输入“End”，不区分字母大小写。正常处理很简单，只需要定义一个字符串，内容为“End”，然后读入字符串并进行比较。关键在于输入错误后清楚缓冲区的处理。

开始时，我是利用getline（）都字符串，长度设置为4。但是会读入‘\n’（不存储），使得缓冲区中不存在‘\n’，如果此时要清除缓冲区，若输入字符串（不含‘\n’）长度大于3，不会有什么影响；但如果小于等于3，由于缓冲区不存在‘\n’，此次并不会清理，这会影响下一次输入。

为了解决这一问题，我决定用getline（）读入所有输入的字符串，然后只比较前三个字符，这样就不用担心清除缓冲区造成的问题。

5. 心得体会

5.1.经验与收获

在本次实验中，我寻找了许多方法来解决各种问题，其中，很多问题并不是只有一种方法可以解决，也有许多问题所找到的方法不一定都正确，还有一些问题不一定能够找到方法，需要去重新审视这个问题，明白自己需要实现什么样的功能，进而将该问题转化为其他问题，以此来寻找解决方案。

在本次作业中，我也更能够将函数按功能拆开，尽量达到各个函数各司其职，既不一个函数负责多个功能，也不多个函数负责一个功能，是正虚的可读性更强，函数利用程度更高。

在本次作业完成的过程中，对于某一程序中的类似部分，我进行了复制粘贴，但忘记了根据具体情况修改参数的值，导致在调试过程中总会出现错误，而自己也很难发现究竟是哪里的错误，致使自己花费了很长时间来调试。因此，对于相似的程序，要复制粘贴的话一定要记着根据情况修改内容，或者直接写而非复制粘贴。

此外，本次程序我也出现了数组越界问题，主要是在进行是否可交换时，直接在9*9的数组上进行讨论，既没有进行特判，也没有扩大数组，从而造成越界。

最后，通过完成本次程序，我学习了很多新的函数，对鼠标相关、用不同颜色输出字符（串）相关的函数有了较为深刻的理解，并能熟练应用。

5.2.总结

我考虑了前后各个程序之间的关联，每一小题均在前一小题的基础上完成，在最后对程序内容进行了修改，既能保证各个程序之间的关联，又能保证各个程序功能的实现。

本次作业我做到了有效利用前面程序的代码，其中利用率较高的时judge（）判断函数，hint（）判断函数、fall（）和fill（）函数等。这些函数是彩球游戏的基本功能，几乎每一项都要求实现，使函数更加简洁，更能体现其基本性，我尽量避免这些函数应参数不同而产生的不同内容，将不同的部分交由外界处理，使这些函数能够被多次调用而基本不产生影响

为了实现代码的更好重用，我们需要深入分析每个功能之间的内在联系。通过仔细分析这些功能，

我们能够发现它们之间的共通点，从而提炼出可以重复使用的部分。这不仅有助于减少代码的冗余，还可以提高代码的可维护性和可扩展性。

在进行功能分析时，我们应重点关注以下几点：

识别共性：找出多个功能中共同的逻辑和操作。将这些共性部分提取出来，形成独立的函数或模块。这些提取出来的部分应该具备高内聚性，即它们内部的功能应该紧密相关，而与外部的耦合应尽量减少。

简洁性和单一职责原则：保持每个函数的简洁性，确保每个函数只负责一个独立的任务。这不仅有助于提高代码的可读性和可测试性，还能使这些函数在不同上下文中被多次调用。

减少参数复杂性：尽量减少函数的参数数量，过多的参数会增加函数的复杂性，降低其重用性。

通过以上方法，我们可以大幅度提高代码的重用性，使其在不同的项目和场景中都能高效地发挥作用。这不仅节省了时间，还能提升代码质量和系统稳定性。

6. 附件：源程序

```
void game(int* array[][9], int row, int column,char order)
{
    int a1[9][9] = { 0 },a2[9][9]={0}, col, line, buffer_col, buffer_line, hang = 2 * row + 5, lie = 40;
    int* pa1[9][9],*pa2[9][9], time = 0, score = 0;
    char str[200], s[] = "END";
    for (int i = 0; i < 9; i++) {
        for (int j = 0; j < 9; j++) {
            pa1[i][j] = &a1[i][j];
            pa2[i][j] = &a2[i][j];
        }
    }
    cct_cls();
    judge(array, row, column, pa1);
    frame(row, column, order, &col, &line, &buffer_col, &buffer_line);
    ball(array, row, column, pa1, order, 1);
    cct_gotoxy(0, 0);
    cout << "屏幕：" << hang << "行" << lie << "列";
    cout << "(当前分数：" << score << " 右键退出)";
    while (1) {
        if (End_judge(a1, time, order)==1) {
            int ret;
            hint(array, row, column, pa2, order);
            ball(array, row, column, pa2, order);
            if(End_judge(a2, time, order) == 1)
                break;
            while (1) {
                cct_gotoxy(14, 0);
                cct_setcolor(COLOR_BLACK, COLOR_WHITE);
                cout << "(当前分数：" << score << " 右键退出)";
                int read_I[2] = { 0 }, read_J[2] = { 0 }, * pread_I[2], * pread_J[2];
                for (int i = 0; i < 2; i++) {
                    pread_I[i] = &read_I[i];
                    pread_J[i] = &read_J[i];
                }
                ret = mouse_read(array, pa2, order, 4, hang, pread_I, pread_J);
                if (ret == 0) {
                    if (mouse_read_judge(array, row, column, read_I, read_J) == -1) {
                        ball(array, row, column, pa2, order);
                        continue;
                    }
                    else if (mouse_read_judge(array, row, column, read_I, read_J) == 1) {
                        cct_showch(0, hang - 3, ' ', COLOR_BLACK, COLOR_WHITE, 40);
                        cct_gotoxy(0, hang - 3);
                        cout<<"不能交换"<< char(65 + read_I[1]) << "行" << read_J[1] + 1 << "列<=>"<< char(65 + read_I[0]) << "行" << read_J[0] + 1 << "列";
                        ball(array, row, column, pa2, order);
                        Sleep(100);
                        continue;
                    }
                    else {
                        int tmp = *array[read_I[0]][read_J[0]];
                        *array[read_I[0]][read_J[0]] = *array[read_I[1]][read_J[1]];
                        *array[read_I[1]][read_J[1]] = tmp;
                    }
                }
                break;
```

```
                        }
                        if (ret == 1)
                                break;
                        judge(array, row, column, pa1);
                        ball(array, row, column, pa1, order, 1);
                        for (int i = 0; i < 9; i++)
                                for (int j = 0; j < 9; j++)
                                        *pa2[i][j] = 0;
                        time++;
                }
                if (time >= 1)
                        score =score + score_counter(pa1, row, column);
                fall(array, row, column, pa1, order);
                fill(array, row, column, pa1, order);
                for (int i = 0; i < 9; i++)
                        for (int j = 0; j < 9; j++)
                                a1[i][j] = 0;
                judge(array, row, column, pa1);
                ball(array, row, column, pa1, order, 1);
        }
        cct_gotoxy(0, 0);
        cout << "无可消除项，游戏结束！最终分数：" << score << ")";
        cct_showch(0, hang - 3, ' ', COLOR_BLACK, COLOR_WHITE, 40);
        cct_gotoxy(0, hang - 3);
        while (1) {
                cout << "本小题结束，请输入End继续...";
                cin.getline(str, 200,'\n');
                if (tj_strcasecmp(str, s) != 0)
                        continue;
                else
                        break;
        }
        cct_setconsoleborder(col, line, buffer_col, buffer_line);
        cct_setfontsize("新宋体", 16, 8);
    }
int mouse_read(int* array[][9], int* pa[][9], char order,int interval,int hang,int* pread_I[2] = { 0 },int* pread_J[2]={0})
{
        cct_enable_mouse();
        int X, Y, I, J,N=0;
        int Action, keycode1, keycode2;
        while (1) {
                int ret=cct_read_keyboard_and_mouse(X, Y, Action, keycode1, keycode2);
                double dI = (Y - 2.0) / (interval / 2);
                double dJ = (X - 2.0) / interval;
                I = int(dI);
                J = int(dJ);
                cct_showch(0, hang - 3, ' ', COLOR_BLACK, COLOR_WHITE, 40);
                cct_gotoxy(0, hang - 3);
                if ((I == dI || I == dI - 0.5) && (J == dJ || J == dJ - 0.25) && (I >= 0 && I <= 8) && (J >= 0 && J <= 8)) {
                        cout <<"[当前光标] "<< char(65 + I) << "行" << J + 1 << "列";
                        if (Action == MOUSE_LEFT_BUTTON_CLICK) {
                                cct_showch(0, hang - 3, ' ', COLOR_BLACK, COLOR_WHITE, 40);
                                cct_gotoxy(0, hang - 3);
                                if (*array[I][J] == *pa[I][J]) {
                                        cout << "当前选择" << char(65 + I) << "行" << J + 1 << "列";
                                        *pread_I[N] = I;
                                        *pread_J[N] = J;
                                        if (N == 1) {
                                                if ((*pread_I[1] - *pread_I[0])* (*pread_I[1] - *pread_I[0])+ (*pread_J[1]- *pread_J[0])*(*pread_J[1] - *pread_J[0])>1) {
                                                        cct_showstr(2 + *pread_J[0] * interval, 2 + *pread_I[0] * interval / 2, "◎", *array[*pread_I[0]][*pread_J[0]], COLOR_BLACK, 1, -1);
                                                        N = 0;
                                                        *pread_I[N] = I;
                                                        *pread_J[N] = J;
                                                }
                                        }
                                        cct_showstr(2 + J * interval, 2 + I * interval / 2, "◎", *array[I][J], COLOR_HWHITE, 1, -1);
                                        N++;
                                        if (order == '8') {
                                                Sleep(500);
                                                break;
                                        }
                                }
                                else
                                        cout << "不能选择" << char(65 + I) << "行" << J + 1 << "列";
                        }
                        else if (Action == MOUSE_RIGHT_BUTTON_CLICK)
                                return 1;
                        if (N == 2)
                                break;
                }
                else
                        cout << "[当前光标] 位置非法";
        }
        cct_disable_mouse();
        return 0;
}

void pause_judge()
{
        while (1) {
                int ret = _getch();
                if (ret == '\n' || ret == '\r')
                        break;
        }
```

```
}

int equal_num(int *array[][9],int row, int column, int *pa[][9])
{
	int ret = 0;
	for (int i = 0; i < row; i++)
		for (int j = 0; j < column; j++)
			if (*pa[i][j] != 0)
				ret++;
	return ret;
}

void hint(int* array[][9],int row,int column ,int* pa[][9],char order)
{
	int A[13][13] = { 0 }, B[13][13] = {0};
	for (int i = 0; i < row; i++)
		for (int j = 0; j < column; j++)
			A[i + 2][j + 2] = *array[i][j];
	for (int i = 2; i < row + 2; i++) {
		for (int j = 2; j < column + 2; j++) {
			if (A[i][j - 1] == A[i][j + 1] && A[i][j - 1] != 0) {
				if (A[i + 1][j] == A[i][j - 1]) {
					B[i + 1][j] = A[i + 1][j];
					B[i][j] = A[i][j];
				}
				if (A[i - 1][j] == A[i][j - 1]) {
					B[i - 1][j] = A[i - 1][j];
					B[i][j] = A[i][j];
				}
			}
			if (A[i - 1][j] == A[i + 1][j] && A[i - 1][j] != 0) {
				if (A[i][j + 1] == A[i - 1][j]) {
					B[i][j + 1] = A[i][j + 1];
					B[i][j] = A[i][j];
				}
				if (A[i][j - 1] == A[i - 1][j]) {
					B[i][j - 1] = A[i][j - 1];
					B[i][j] = A[i][j];
				}
			}
			if (A[i][j - 2] == A[i][j - 1] && A[i][j - 1] != 0) {
				if (A[i][j + 1] == A[i][j - 1]) {
					B[i][j + 1] = A[i][j + 1];
					B[i][j] = A[i][j];
				}
				if (A[i+1][j] == A[i][j - 1]) {
					B[i + 1][j] = A[i+1][j];
					B[i][j] = A[i][j];
				}
				if (A[i-1][j] == A[i][j - 1]) {
					B[i - 1][j] = A[i-1][j];
					B[i][j] = A[i][j];
				}
			}
			if (A[i][j + 2] == A[i][j + 1] && A[i][j + 1] != 0) {
				if (A[i][j - 1] == A[i][j + 1]) {
					B[i][j - 1] = A[i][j -1];
					B[i][j] = A[i][j];
				}
				if (A[i + 1][j] == A[i][j + 1]) {
					B[i + 1][j] = A[i + 1][j];
					B[i][j] = A[i][j];
				}
				if (A[i - 1][j] == A[i][j + 1]) {
					B[i - 1][j] = A[i - 1][j];
					B[i][j] = A[i][j];
				}
			}
			if (A[i-2][j] == A[i-1][j] && A[i-1][j] != 0) {
				if (A[i+1][j] == A[i - 1][j]) {
					B[i + 1][j] = A[i+1][j];
					B[i][j] = A[i][j];
				}
				if (A[i][j+1] == A[i - 1][j]) {
					B[i][j + 1] = A[i][j + 1];
					B[i][j] = A[i][j];
				}
				if (A[i][j-1] == A[i - 1][j]) {
					B[i][j - 1] = A[i][j - 1];
					B[i][j] = A[i][j];
				}
			}
			if (A[i + 2][j] == A[i + 1][j] && A[i + 1][j] != 0) {
				if (A[i - 1][j] == A[i + 1][j]) {
					B[i - 1][j] = A[i - 1][j];
					B[i][j] = A[i][j];
				}
				if (A[i][j + 1] == A[i + 1][j]) {
					B[i][j + 1] = A[i][j + 1];
					B[i][j] = A[i][j];
				}
				if (A[i][j - 1] == A[i + 1][j]) {
					B[i][j - 1] = A[i][j - 1];
					B[i][j] = A[i][j];
				}
			}
		}
	}
	for (int i = 0; i < row; i++)
		for (int j = 0; j < column; j++)
			*pa[i][j] = B[i + 2][j + 2];
}
```

```
void fall(int *array[][9],int row,int column, int* pa[][9],char order)
{
	int ret_array;
	cout << endl;
	if (order >= '7')
		climinate(array, row, column, pa);
	for (int i = 0; i < column; i++) {
		for (int j = 0; j < row; j++) {
			if (*array[j][i] == *pa[j][i]) {
				*array[j][i] = 0;
				*pa[j][i] = 0;
			}
		}
	}
	for (int i = 0; i<column ; i++) {
		for (int j = 0; j <row; j++) {
			for(int k =0;k<row-j-1;k++)
			if (*array[k][i] != 0&& *array[k+1][i]==0) {
				if (order >= '7')
					cartoon_fall(array, pa, i, k);
				ret_array = *array[k][i];
				*array[k][i] = *array[k + 1][i];
				*array[k + 1][i] = ret_array;
			}
		}
	}
}

void fill(int* array[][9], int row, int column, int* pa[][9],char order)
{
	cout << endl;
	srand((unsigned)time(NULL));
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < column; j++) {
			if (*array[i][j] == *pa[i][j]) {
				*array[i][j] = rand() % 9 + 1;
				*pa[i][j] = *array[i][j];
				if (order >= '7')
					pad(array, pa, j, i);
			}
		}
	}
}

void judge(int* array[][9], int row, int column, int* pa[][9])
{
	int num;
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < column; j++) {
			int k = j + 1;
			num = 1;
			for (; k < column; k++) {
				if (*array[i][k] == *array[i][j])
					num++;
				else
					break;
			}
			if (num >= 3)
				for (int n = j; n < j + num; n++)
					*pa[i][n] = *array[i][n];
			j += (num - 1);
		}
	}
	for (int i = 0; i < column; i++) {
		for (int j = 0; j < row; j++) {
			int k = j + 1;
			num = 1;
			for (; k < row; k++) {
				if (*array[k][i] == *array[j][i])
					num++;
				else
					break;
			}
			if (num >= 3)
				for (int n = j; n < j + num; n++)
					*pa[n][i] = *array[n][i];
			j += (num - 1);
		}
	}
   }
void climinate(int* array[][9], int row, int column, int* pa[][9])
{
	int interval = 4;
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < column; j++) {
			if (*array[i][j] == *pa[i][j]) {
				cct_showstr(2 + j * interval, 2 + i * interval / 2, "□", *array[i][j], COLOR_BLACK, 1, -1);
				Sleep(100);
				cct_showch(2 + j * interval, 2 + i * interval / 2, ' ', COLOR_HWHITE, COLOR_BLACK,  2);
				Sleep(100);
			}
		}
	}
	cct_setcolor(COLOR_BLACK, COLOR_WHITE);
}

void cartoon_fall(int* array[][9], int* pa[][9],int X,int Y)
{
	int interval = 4;
	cct_showch(2 + X * interval, 2 + Y * interval / 2, ' ', COLOR_HWHITE, COLOR_BLACK, 2);
	cct_showstr(2 + X * interval, 2 + (Y+1) * interval / 2, "○", *array[Y][X], COLOR_BLACK, 1, -1);
```

```
	Sleep(100);
	cct_setcolor(COLOR_BLACK, COLOR_WHITE);
}

void pad(int* array[][9], int* pa[][9], int X, int Y)
{
	int interval = 4;
	cct_showstr(2 + X * interval, 2 + Y * interval / 2, "○", *array[Y][X], COLOR_BLACK, 1, -1);
	Sleep(100);
	cct_setcolor(COLOR_BLACK, COLOR_WHITE);
}

void ball(int* array[][9], int row, int column, int* pa[][9], char order,int ins)
{
	int interval[2] = { 2,4 }, N;
	if (order == '4' || order == '6')
		N = 0;
	else
		N = 1;
	for (int i = 0; i < row; i++) {
		for (int j = 0; j < column; j++) {
			if (*array[i][j] == *pa[i][j]) {
				if(ins==1)
				    cct_showstr(2 + j * interval[N], 2 + i * interval[N] / 2, "●", *array[i][j], COLOR_BLACK, 1, -1);
				else
					cct_showstr(2 + j * interval[N], 2 + i * interval[N] / 2, "◎", *array[i][j], COLOR_BLACK, 1, -1);
			}
			else
			    cct_showstr(2+j* interval[N], 2+i* interval[N]/2, "○", *array[i][j], COLOR_BLACK, 1, -1);
			jx(order);
		}
	}
	cct_setcolor(COLOR_BLACK, COLOR_WHITE);
}

void frame(int row, int column, char order,int*pcol,int *pline,int *pbuffer_col,int *pbuffer_line)
{
	int interval[2] = { 2,4 }, N;
	cct_getconsoleborder(*pcol,*pline,*pbuffer_col,*pbuffer_line);
	int hang[2] = { row + 6,2 * row + 6 }, lie = 40;
	if (order == '4' || order == '6')
		N = 0;
	else
		N = 1;
	cct_setconsoleborder(lie, hang[N], lie, hang[N]);
	cct_setfontsize("新宋体", 28, 14);
	cct_showstr(0, 1, "┌", COLOR_HWHITE, COLOR_BLACK, 1, -1);
	jx(order);
	for (int i = 0; i < column; i++) {
		cct_showstr(2 + interval[N] * i, 1, "—", COLOR_HWHITE, COLOR_BLACK, 1, -1);
		jx(order);
		if (order != '4' && order != '6'&& i + 1 < column) {
			cct_showstr(interval[N] * (i + 1), 1, "┬", COLOR_HWHITE, COLOR_BLACK, 1, -1);
			jx(order);
		}
	}
	if(order != '4' && order != '6')
		cct_showstr(interval[N] * column, 1, "┐", COLOR_HWHITE, COLOR_BLACK, 1, -1);
	else
		cct_showstr(2+interval[N] * column, 1, "┐", COLOR_HWHITE, COLOR_BLACK, 1, -1);
	jx(order);
	for (int i = 0; i < row; i++) {
		cct_showstr(0, 2 + i * interval[N] / 2, "│", COLOR_HWHITE, COLOR_BLACK, 1, -1);
		for (int j = 0; j < column; j++) {
			cct_showch(2 + interval[N] * j, 2 + i* interval[N] / 2, ' ', COLOR_HWHITE, COLOR_BLACK, 2);
			jx(order);
			if (order != '4' && order != '6') {
				cct_showstr(interval[N] * (j + 1), 2 + i * interval[N] / 2, "│", COLOR_HWHITE, COLOR_BLACK, 1, -1);
				jx(order);
			}
		}
		if (order == '4' || order == '6') {
			cct_showstr(interval[N] * column + 2, 2 + i * interval[N] / 2, "│", COLOR_HWHITE, COLOR_BLACK, 1, -1);
			jx(order);
		}
		if (i + 1 < row) {
			if (order != '4' && order != '6') {
				cct_showstr(0, 3 + i * interval[N] / 2, "├", COLOR_HWHITE, COLOR_BLACK, 1, -1);
				jx(order);
				for (int j = 0; j < column; j++) {
					cct_showstr(2 + interval[N] * j, 3 + i * interval[N] / 2, "—", COLOR_HWHITE, COLOR_BLACK, 1, -1);
					jx(order);
					if (j + 1 < column) {
						cct_showstr(interval[N] * (j + 1), 3 + i * interval[N] / 2, "┼", COLOR_HWHITE, COLOR_BLACK, 1, -1);
						jx(order);
					}
				}
				cct_showstr(interval[N] * column, 3 + i * interval[N] / 2, "┤", COLOR_HWHITE, COLOR_BLACK, 1, -1);
				jx(order);
			}
		}
	}
	if (order == '4' || order == '6') {
		cct_showstr(0, 2 + row * interval[N] / 2, "└", COLOR_HWHITE, COLOR_BLACK, 1, -1);
		jx(order);
		for (int i = 0; i < column; i++) {
			cct_showstr(2 + i * interval[N], 2 + row * interval[N] / 2, "—", COLOR_HWHITE, COLOR_BLACK, 1, -1);
			jx(order);
		}
		cct_showstr(2 + interval[N] * column, 2 + row * interval[N] / 2, "┘", COLOR_HWHITE, COLOR_BLACK, 1, -1);
		jx(order);
	}
```

```
    else {
            cct_showstr(0, 1 + row * interval[N] / 2, "└", COLOR_HWHITE, COLOR_BLACK, 1, -1);
            jx(order);
            for (int i = 0; i < column; i++) {
                    cct_showstr(2 + i * interval[N], 1 + row * interval[N] / 2, "—", COLOR_HWHITE, COLOR_BLACK, 1, -1);
                    jx(order);
                    if (i + 1 < column) {
                            cct_showstr(interval[N] * (i + 1), 1 + row * interval[N] / 2, "┴", COLOR_HWHITE, COLOR_BLACK, 1, -1);
                            jx(order);
                    }
            }
            cct_showstr(interval[N] * column, 1 + row * interval[N] / 2, "┘", COLOR_HWHITE, COLOR_BLACK, 1, -1);
            jx(order);
    }
}
```