同济大学 实验报告

装 订 线

高级语言程序设计实验报告

——彩球游戏

姓　　名：杨恺铭

学　　号：2152988

专　　业：计算机科学与技术

完成日期：2022.12.18

1. 题目：彩球游戏

1.1内部数组版：

输入行列后，在规定范围内随机生成5个球的位置，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

起始位置必须有球，目的位置必须为空

生成过程中，如果该位置已经有球，要重新生成

球的位置用不同颜色标出

连续5个则消除，并可以得分(规则可以自定义，demo的规则是消除数量为n，则得分为(n-1)*(n-2)，和之前的游戏并不相同，双五连等情况，交叉点要重复计数)

本次移动若得分，则不产生新球，否则会产生三个新球

没有任何空位则游戏结束

1.2伪图形界面

cmd伪图形界面上画出框架(无分隔线)及初始的五个球

demo程序中设置字体、屏幕大小等操作从cmd_ console_ tools 中寻找

菜单项5:在cmd伪图形界面上画出框架(有分隔线)及初始的五个球

要求同菜单项4.

菜单项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，完成一一个移动

鼠标操作为:左键选择，右键退出本小题

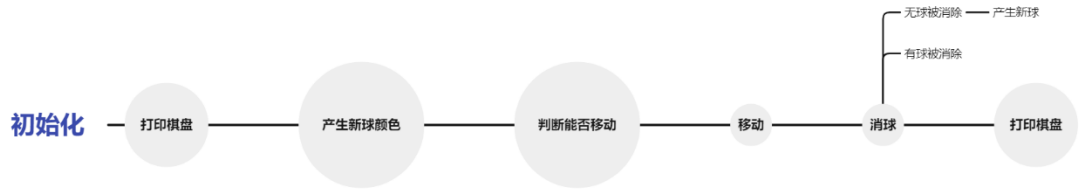
鼠标移动过程中，要实时显示当前移动到n*n矩阵的哪个位置(行：A-I，列：1-9)， 放在边框线上不算

移动过程需要完整的移动轨迹显示，动画效果必须跨越分隔线

菜单项7:在cmd伪图形界面上实现完整版的游戏

2. 整体设计思路

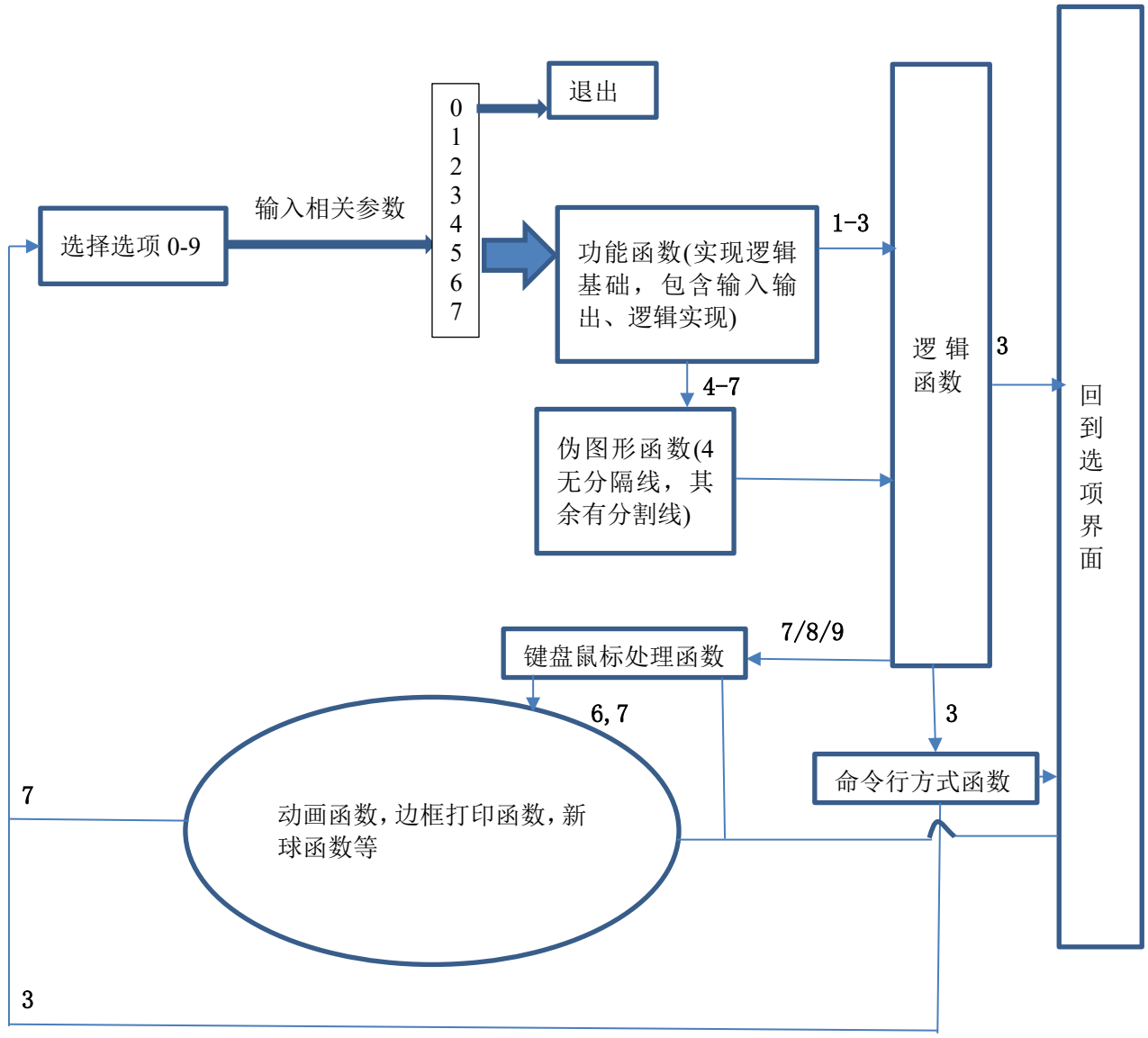
整体设计思路应该从选项3开始



上图是选项3的基本流程图。由于行列数最大值为9，为防止溢出将二维数组大小定义为9×9。根据逻辑分析，仅仅一个二位数组并不能实现相邻相同值寻找记录并合并的过程，据此定义一个所有值都为0（字符0），大小也是9×9的字符数组，来配合记录数据的二维数组实现遍历、转化、更新等具体功能。当需要改变二维数组特定几个值时，将需要改变的值在字符数组中用*号标出，遍历整个字符数组，当读到*号时在数值数组的相同位置改变其数值。

伪图形的逻辑则是根据不同颜色来造成视觉上的图形效果。为了使背景布足够明显，将其颜色设置为亮黑色，边框线设置为黑色，边框线所用符号取自word中的宋体符号库。中间数据呈现部分则是根据数值将其输入取颜色函数，获得不同的颜色，宽度6列，高度3行，外围一圈是边框，中间是数字。至此游戏界面得以完全呈现。伪图形过程则根据数组的交换，数值为0的位置往上走时，对应的两个伪图形模块进行交换，伪图形色块最上面一行重新输出为亮白色，将色块下面的第一行输出为色块颜色，再加上延时可实现动画效果，重复两次即可实现伪图形色块的移动。游戏版则基于上述设计思路通过循环的思路实现。

3. 主要功能的实现



装 订 线

4. 调试过程碰到的问题

1、cct_gotoxy()中x对应列，而y对应行。但是在编写过程中确没有仔细理清楚，前半段的xy和后半段的row col不一致，但前半部分逻辑居然闭环了。由于时间紧迫，无法推翻重写，而是在传入数据的时候颠倒一下顺序，最后反复测试后没有发现异常。

2、寻找路径且最短的方法就是BFS，需要用到自己实现的队列，不多赘述。但在想如何消除双五连确实碰到了问题。当时刚好晚上有别的ddl，心情非常烦躁，觉得自己肯定想不出来。但是睡了个午觉爬起来一想就想通了。双五连和单五连的区别就是交叉点被重复利用。那如何解决重复利用的问题呢？只需要把每一步判断都用单五连来完成，但是找到后并不直接消除，而是等所有找完再统一消除即可。想不到方法不要死磕，而是应该先缓一缓，换个脑子。

3、会发现score分数会莫名其妙的增加，而分数改变的语句只有一个。按理说其他地方不会改变分数。分数统一放在结构体里面。最后发现是vertex[]越界访问造成的加一。

```
struct game {
    int board[ROW_MAX][LINE_MAX];
    int mark[ROW_MAX][LINE_MAX];
    int row, line;
    int vertex_num= 0;
    int vertex[COLOR_MAX + 1] = { 0 };
    int score = 0;
    int vertex_bingo[COLOR_MAX + 1] = { 0 };
};
```

Vertex只有COLOR_MAX个，但由于颜色是1-7，所以数组应该加一才对。所以vertex越界访问了score。没想到检查严格的VS也发现不了这里的问题，只能说这是一个经验的积累吧。

4、如何让鼠标只读取框内（边界时不读入）也是个新奇的点。按数学思想应该是算出哪些位置是有效的，然后比较判断。但本题有效区域都是离散的，而且很多，难以用数学公式归纳概括。而如果使用计算机思维，可以用笨办法。遍历每一个有效区域，如果和鼠标位置相同，则读入，反之不读。也就是说，使用鼠标时，程序在不断的循环遍历每一个方框。这样实现简洁有效。

5. 心得体会

5.1. 这次作业我花费的总时间大概是36小时左右，包括从读题开始到写代码、调试程序，最后改bug，经过这个过程我有不少心得体会。从知识层面上我学会了一些新的知识。首先，在cmd窗口使用鼠标是我第一次接触的东西。研究了老师给的读入鼠标的代码和逻辑，明白了返回鼠标坐标值、鼠标action且同时能够读到键盘的方法。其次，尽管老师给的函数有一部分我并未用到，但我仍通过阅读代码并尝试学会了一些新的知识。取当前光标所在位置的坐标值、设置光标状态（显示/不显示/全高/半高/横线等）、允许使用鼠标、取消允许使用鼠标，这些都是我第一次接触到的知识。至于经验教训，我明白了如果遇到一个bug或难题迟迟不能更正，则可以先跳过这一部分去进行下一部分代码的编写，在写下一部分的时候自己很可能会得到启发，用后续的思考来改正之前的代码，这样效率会提高不少，还能够不上网搜索，而是通过自己的尝试找到解决问题的方法。

5.2.自顶向下的程序设计方法也很重要。我也更清楚的理解编程只是一种手段，他做的是把脑子里的实现方法，用编程传递给电脑。我可以通过这些小题掌握一定的方法与途径，在做后续小题以及汇总时可以更快，思路也更加清晰。如：第一题和第二题是最基本的数组实现以及相同值的寻找。实现这个功能还需要一个辅助的数组来记录寻找到的相连相同值，如果让我直接做伪图形动画，我可能很难想到这样的解决方式。另外，在整合为最终版本时，输出逻辑部分可以直接应用之前选项的内容，仅需整合并调试即可，逻辑部分是完全相同的。前序选项的处理为大作业的进行提高了效率，提供了逻辑和思路，在整合时更加方便快捷，避免了很多不必要的失误，能够有效减少bug数量，使得整体完成过程更加流畅。

5.3. 与汉诺塔大作业相比，这是第一次独自封装函数。但是每个函数不超过50行实在是有些困难。也不是说不能进一步封装，而是觉得封装后意义不大。所以导致某几个函数（特别是鼠标控制函数）超过了一百行，这里还是需要改进的。我在完成过程能够更加充分地考虑了前后小题的关联关系，尽可能做到后面小题有效利用前面小题已完成的代码，避免对代码进行简单的复制粘贴，提高代码利用率，减少代码行数。

5.4. 至于编写一个复杂程序应当注意的，是在应当将更多代码封装成函数的形式并将函数命名为与这一段代码相关的名称（便于理清思路）。函数可以随意调用，避免编写重复代码，可以提高编程效率，也更容易理清逻辑和思路，改bug时可以根据错误点找到相应的函数，对函数进行阅读并需找错误将其修改而不必对全部代码进行阅读，能够迅速定位，节省很多时间。函数的编写对于复杂程序来说十分重要，函数的运用对于节省代码、节省时间、提高效率具有很高的价值和意义。例如此次合成十游戏大作业我在90-b2.h里共声明了二十多个函数，main函数、console函数、tools函数、base函数只需直接调用即可，很大程度上节省了空间，使代码量大幅减少，同时又思路清晰，便于检查错误。

6. 附：源程序

装

订

线

```
//打印棋盘里的球
void prt_ball_gui(game& G)
{
    for (int y = 0; y < G.row; y++) {
        for (int x = 0; x < G.line; x++) {
            if (G.board[y][x] != 0) {
                cct_showstr(POINT_X + x * 4, POINT_Y + y * 2,
                    "○", G.board[y][x], FG);
            }
            else {
                cct_showstr(POINT_X + x * 4, POINT_Y + y * 2,
                    "  ", BG, FG);
            }
        }
    }
}
int if_avai_area(const game& G, int X, int Y, int & cur_x,int &cur_y)
{
    int ret = 0;
    for (int i = 0; i < G.row; i++) {
        for (int j = 0; j < G.line; j++) {
            if ((Y == BOARDER_Y + i * 2 + 1)
                && ((X == BOARDER_X + 2 + j * 4) || (X == BOARDER_X + 2 + j * 4 + 1))) {
                ret = 1;
                cur_x = j;
                cur_y = i;
                return ret;
            }
        }
    }
    return ret;
}

//超大的bug，还不敢随便改
void trans_xy(point& p)
{
    int y1 = p.x, x1 = p.y;
    p.x = x1;
    p.y = y1;
}
void prt_move(game& G, point src, point dst,int bg)
{
    const int S = 300;//延时
    trans_xy(src);
    trans_xy(dst);
    //擦除
    cct_showstr( POINT_X + src.x * 4, POINT_Y + src.y * 2,
        "  ", BG, FG);
```

```
    //间接移动，取中点
    cct_showstr((POINT_X + src.x * 4 + POINT_X + dst.x * 4) / 2,
        (POINT_Y + src.y * 2 + POINT_Y + dst.y * 2)/2,
        "◎", bg, FG);
    Sleep(S);
    if (src.x == dst.x) {
        cct_showstr((POINT_X + src.x * 4 + POINT_X + dst.x * 4) / 2,
            (POINT_Y + src.y * 2 + POINT_Y + dst.y * 2) / 2,
            "═", BG, FG);
    }
    else {
        cct_showstr((POINT_X + src.x * 4 + POINT_X + dst.x * 4) / 2,
            (POINT_Y + src.y * 2 + POINT_Y + dst.y * 2) / 2,
            "‖", BG, FG);
    }
    //最终移动tobe
    cct_showstr( POINT_X + dst.x * 4, POINT_Y + dst.y * 2,
        "◎", bg, FG);
    Sleep(S);
    cct_setcolor();

    }



void game_gui(game& G)
{
    //清空并开始游戏
    G.score = 0;
    clear_G_mark(G);

    cct_enable_mouse();    //打开鼠标
    cct_setcursor(CURSOR_INVISIBLE);    //关闭光标


    const int NEW_BALL = 3;
    int next_color[NEW_BALL] = { 0 };
    int flag_new = 1;
    point src = { -1,-1 };
    while (1) {
        if (G.vertex_num >= G.row * G.line) {
            cct_gotoxy(0, BOARDER_Y + G.row * 2 + 1);
            cout << "无空位可移动，游戏结束!" << endl;
            break;
        }
        //产生新颜色
        generate_color(next_color, NEW_BALL, flag_new);
        //打印新颜色
        prt_new(next_color, NEW_BALL);

        int X = 0, Y = 0, cur_x, cur_y;//cur_x是列
        int ret, maction;
        int keycode1, keycode2;
```

```
        int loop = 1;

        while (loop) {
            ret = cct_read_keyboard_and_mouse(X, Y, maction, keycode1, keycode2);
            if (ret == CCT_MOUSE_EVENT) {
                if (if_avai_area(G, X, Y, cur_x, cur_y)) {
                    cct_gotoxy(0, BOARDER_Y + G.row * 2 + 1);
                    cout << "[当前光标]" << char(cur_y + 'A') << "行" << cur_x + 1 << "列
";
                    cout << setw(20) << "";
                    int quit = 0;
                    switch (maction) {
                        case MOUSE_ONLY_MOVED:
                            break;
                        case MOUSE_LEFT_BUTTON_CLICK:           //按下左键
                            if (src.x == -1 && src.y == -1) {
                                if (G.board[cur_y][cur_x] != 0) {
                                    src.x = cur_x;
                                    src.y = cur_y;
                                    cct_showstr(POINT_X + src.x * 4, POINT_Y + src.y *
2,
                                        "◎", G.board[src.y][src.x], FG);
                                    cct_setcolor();

                                }
                            }
                            else {
                                if (G.board[cur_y][cur_x] != 0) {
                                    cct_showstr(POINT_X + src.x * 4, POINT_Y + src.y *
2,
                                        "○", G.board[src.y][src.x], FG);
                                    cct_setcolor();
                                    src.x = cur_x;
                                    src.y = cur_y;
                                    cct_showstr(POINT_X + src.x * 4, POINT_Y + src.y *
2,
                                        "◎", G.board[src.y][src.x], FG);
                                    cct_setcolor();

                                }
                                else {
                                    point dst = { cur_x,cur_y };
                                    stack path_s;
                                    point dst1 = dst, src1 = src;
                                    trans_xy(dst1);
                                    trans_xy(src1);
                                    if (find_path(G, src1, dst1, path_s)) {
                                        cct_gotoxy(0, BOARDER_Y + G.row * 2 + 1);
                                        cout << "[提示] 可以从" << char(src.y + 'A')
                                            << src.x + 1 << "移动到"
                                            << char(dst.y + 'A') << dst.x + 1;
                                        cout << setw(14) << "";
                                        //移动过程打印
                                        int color = G.board[src1.x][src1.y];
```

装 订 线

```
for (int i = path_s.top - 1; i > 0; i--) {
    prt_move(G, path_s.base[i], path_s.base[i - 1], color);
}

//默认产生新球
flag_new = 0;

//swap
int tmp = G.board[src1.x][src1.y];
G.board[src1.x][src1.y] = G.board[dst1.x][dst1.y];
G.board[dst1.x][dst1.y] = tmp;
src = dst;

//游戏部分仔细商榷

//消除球
int bingo_n = bingo(G);
//未消除成功
if (!bingo_n) {
    //产生新球
    flag_new = 1;
    gen_ball(G, next_color, NEW_BALL);
    prt_ball_gui(G);
    //显示被选中的球

    cct_showstr(POINT_X + src.x * 4, POINT_Y + src.y * 2,
        "◎", G.board[src.y][src.x], FG);
    cct_setcolor();

}
else {//消除成功
    //更新分数
    cct_gotoxy(OTHER_X + 2 + 6, OTHER_Y + 1);
    cct_setcolor(BG, FG);
    cout << G.score;
    cct_setcolor();

    //状态机回到初始值
    src.x = -1;
    src.y = -1;
    prt_ball_gui(G);
}



//更新当前比例
prt_per(G);
//option6
quit = 1;
break;
```

```
                                    }
                                    else {

                                        cct_gotoxy(0, BOARDER_Y + G.row * 2 + 1);
                                        cout << "[错误] 无法从" << char(src.y + 'A')
                                             << src.x + 1 << "移动到"
                                             << char(dst.y + 'A') << dst.x + 1;
                                        cout << setw(14) << "";


                                    }
                                    clear_G_mark(G);
                                }
                            }


                            break;
                        case MOUSE_RIGHT_BUTTON_CLICK: //按下右键
                            return;
                        default:
                            break;
                    }
                    //option6
                    if (quit) {
                        break;
                    }
                }
            }

        }

    }

cct_disable_mouse();  //禁用鼠标
    cct_setcursor(CURSOR_VISIBLE_NORMAL);   //打开光标

}
```