高级程序语言实验报告

---彩球游戏

装

订

线

姓名：童佳燕

学号：1551445

班级：计算机科学与技术1班

指导教师：沈坚

完成日期：2016.1.8

1.实验题目：彩球游戏实现过程

1.1 问题描述（游戏规则）：

（1）游戏区域为 7*7-9*9(具体可由键盘输入确定)，共有七种颜色的彩球随机出现，其中状态为 5 个，以后每次出现 3 个。
（2）用鼠标选中某个彩球，在选择一个空白区域作为目标位置，如果从源到目标位置有通路可走，将彩球移动到目标位置；如果没有通路可走，则不移动并给出提示。
（3）当同色彩球在横向，纵向，斜向达到 5 个及以上时，可以消除，同时得到相应的分数。
（4）当棋盘中没有空位时，游戏结束。

1.2 题目要求：

1.2.1 整体要求：

(1)所有小题放在一个程序中，以菜单的形式进行选择
(2)右侧得分，预告彩球以及统计信息。
(3)用伪图形界面进行游戏设计。

1.2.2 显示要求：

（1）被选中的彩球要有不同效果
（2）彩球移动是，要有动画效果沿着通路进行移动
（3）消除时要有相应的动画效果
（4）打印球时以不同的颜色输出
（5）伪图形界面中，鼠标在棋盘上移动实时显示当前所在的行，列位置。

1.2.3 游戏规则具体要求：

（1）连续 5 个及以上则消除，得分规则自定义（具体实现中，我的规则是消除数量为 n，则得分为（n-1）*(n-2)，其中交叉点要重复计数）。
（2）若本次移动得分，则不产生新球，否则随机产生三个新球。
（3）鼠标左键选择，右键退出。
（4）小彩球只允许在空白位置进行移动，且只可横或竖走，不允许对角线走。

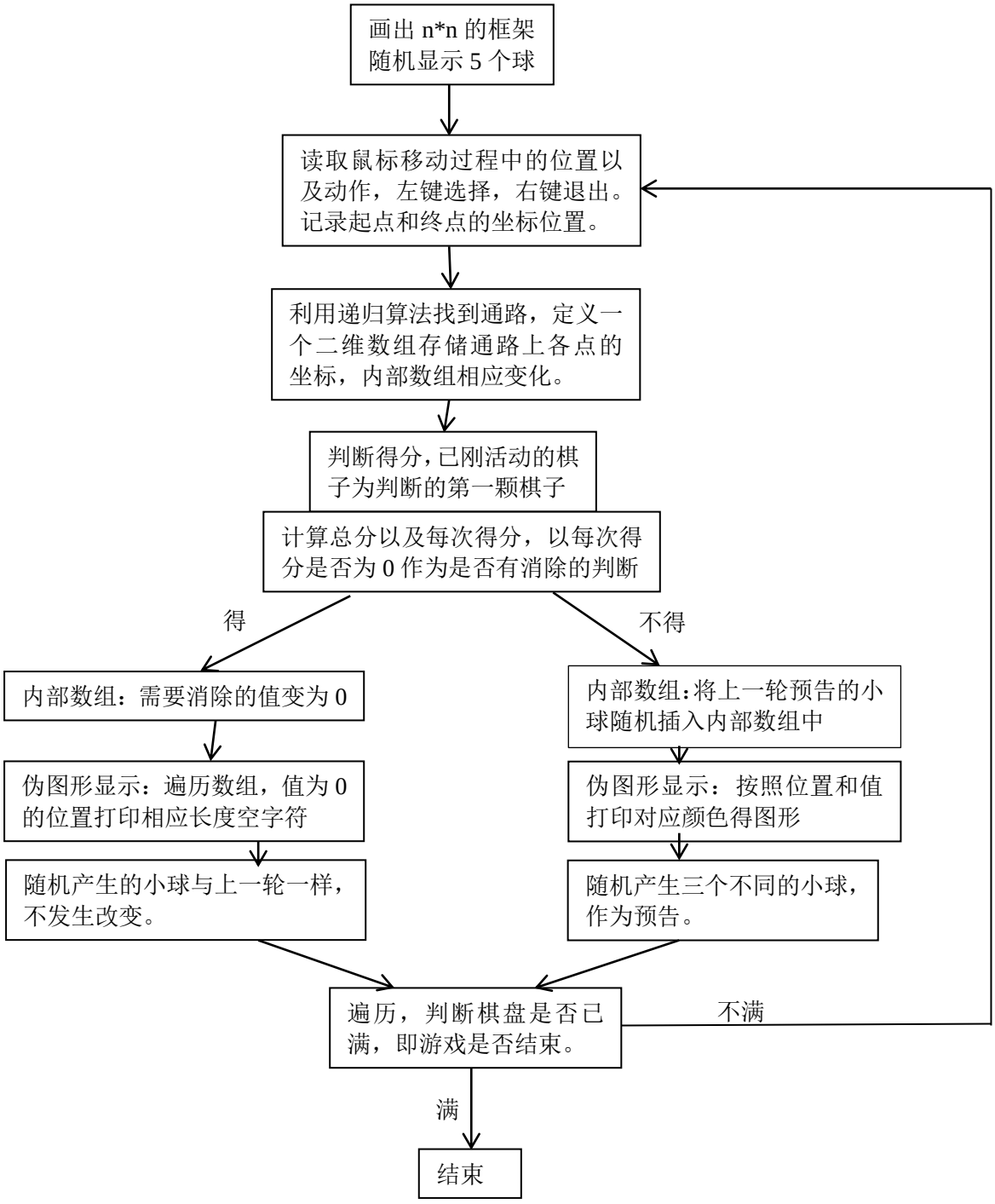
1.2.4 代码要求

（1）尽量使各菜单项中的程序公用函数，用参数解决细微差异。
（2）各函数代码长度尽量不超过 50 行
（3）不允许使用全局变量，全局指针以及全局数组，允许使用宏定义。
（4）只允许使用目前为止讲过的内容（包括课后补充知识。）

2. 整体设计思路

写这个程序首先要考虑的是它的本质，也就是数据在机器内的存储形式。考虑多方面因素，最合适的应该是二维数组，形象的可以看成是 10*10 的行列矩阵（二维数组事实上还是连续排列的数据存储，这里为了方便理解）。其中数组的元素的值代表不同颜色的小球，0 代表空位。程序设计过程中，可以通过数组相应位置（棋盘相应位置）的数值进行是否有球，是否同色等的判断，同时当球移动时，也可以通过移动路径中相应位置的值的变化来实现，同样包括当判断五个相同颜色的小球同线是消去的实现。

整体的过程如下图：



装 订 线

3. 主要功能的实现

3.1 寻路

3.1.1 利用递归算法找到通路

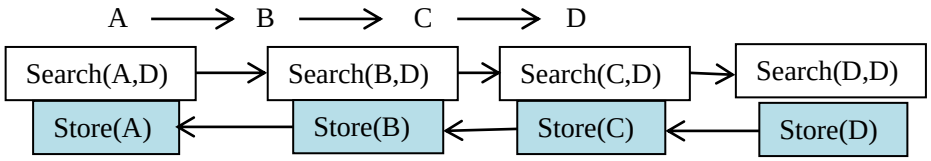
自定义函数找到鼠标左键选择的起点和终点后，将起点和终点的坐标（即在二维数组内部存储的 i,j 值）存入两个大小为 2 的数组 begin[2]和 final[2].

确定好起点和终点后，就可以开始寻路了。将整一个过程拆分成多个小过程，每个过程由两步组成----起点走一步到起点 2；起点 2 走到终点。不需要考虑每一个小过程到底发生了什么。

函数返回类型设置为 int 型，0 代表无返回，也就是没找到路；1 代表找到路。

找到之后原路返回，一路 return 同时存储路径上的点 的坐标。算法完成之后，路径上所有的点逆序存放在 rout 数组之中。

例如下图：



3.1.2 找通路时内部存储变化

每次查找过程中，首先判断是否为结束条件，如果不是，则将内部数组该位置上的数值设置为-1，避免重复判断查找，保证不走同一个位置。而在判断能不能走的条件时，首先，要走的点对应的数值需要为 0，同时不超出棋盘界限，所以当 search 完之后，需要将所有值为-1 的恢复成为原先的值，也就是 0.

除此之外，起点和终点的值要进行互换。

不是所有值为-1 的都是路径上的点，还有可能是探了一下但没有成功的点！ 真正路径上的点都存储在相应的数组中。通过路径数组中存储的位置值，也就是对应二维数组中的 i,j 值对输出形式，位置进行选择。

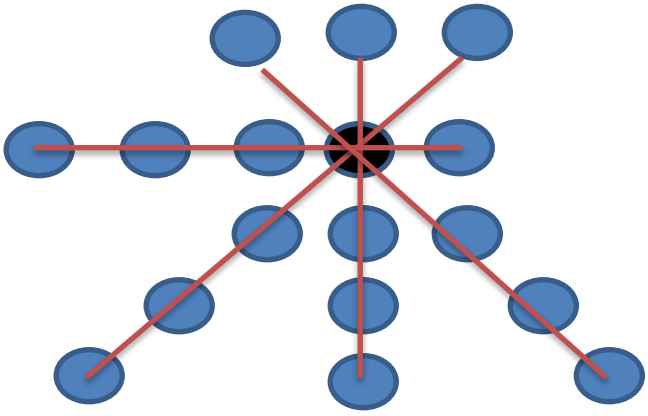
3.1.3 伪图形显示移动路径

通过路径数组中存储的位置值，也就是对应二维数组中的 i,j 值确定路径!由于存储时是先存终点后存起点，所以要逆方向读取。

首先需要判断路径上有几个点，用简单的循环和条件就可以判断。其次从倒数第二个开始，对应伪图形界面中的坐标，打印相应颜色的字符串，同时在倒数第一个位置打印空字符串，以此类推，一直到路径数组中的点全都对应完毕。

3.2 判断得分

在新的一颗子落下之前，没有达到可以消除的条件；这颗子落下后，消除条件形成，发生消除动作。所以，每次寻路成功之后，以此次寻路中的终点作为第一颗子进行展开，判断是否满足消除条件。



（如图只是为解释，该情况在实际中是不可能出现的）

判断时，从黑色小球出发，以上下为例，分两个方向进行搜索，分别定义 i=1,j=1,循环判断 p[y-i][x]?=p[y][x]，不等于循环结束，记录 i 值；等于则 i++，继续搜索。。。j 同理，向黑色小球的下方进行搜索，最终纵向的小球个数为 i+j-1;其他方向都是同理。

由于交叉点需要重复计数，所以选择先判断完所有方向，记录所有可以消除的球（包括交叉点的多次计数）后再对内部数组进行改变，防止交叉小球满足一个条件后就被消除，对潜在的其他方向的消除条件判断造成影响。

记录完之后计算此次得分，以此次得分是否为零，用以后续判断是否进行了消除。

4. 调试过程中遇到的问题

4.1 寻路过程

在利用递归寻路过程中，一直爆栈！原因是由于前面在输入起点和终点的时候已经对点的合法性做了判断，以为在递归是就不需要了！所以没有加上点坐标不能超过行列的限制条件，导致不断地爆栈。而事实上，输入起点和终点时的判断只是保证了最初的起点和终点的合法性，却不能保证之后再每一步递归的过程中的合法性。

在加上限制条件后，寻路算法正确。

4.2 伪图形界面坐标的确定

整个程序有多套变量相关于伪图形界面输出的坐标，比如输入的起点和终点的坐标 begin,final 数组中 A-I,1-9，以及在数组中存储的 0-8，还有在输出伪图形界面时一开始为了方便判断，定义的，第几个字符串，以及鼠标移动读取的坐标的值。如果时间充足，把这些参数统一一下会更好。

4.3 小球只能走一次

具体情况是第六小题，第一次执行正确，第二次执行错误，感觉有什么数据在一次执行之后没有被释放，第二次执行继承了之前的数据。对于第七小题，就是第一步执行正确，第二步及以后，都是错误的！基本路线以及规则还是正确的，只是小球会“自己长脚”跑掉，步数不定。

经过老师的指点，原因是在寻路函数中为了记录步数来对路径数组赋值时使用了静态变量，导致两次选择同一个菜单时，静态变量存储的数据还没有被释放，还保留着原来的路径的点。同样导致第七小题无法正常进行。

解决方法是在给寻路函数添加了一个指针参数，指向函数外的一个变量，每次执行参数都会被重置为0。这样就解决了“被继承”的尴尬！

收获颇大，对于静态变量，使用时要小心，对于前后一直有关联的，用静态比较方便，但如果需要重置，则会比较麻烦。因为数据如果放在函数里面，只有函数自己去改变，外面无法访问，尤其是对于递归函数来说，不能在每次最后重置。静态变量只有在执行文件关掉后才会被释放。

4.4 小球吃小球

在具体执行过程中（菜单项 7），偶尔会出现小球移动过程中直接把另一个小球吃掉，也就是没有发现阻碍，“畅通无阻”。

本来猜想可能是把前一局预告的小球数据插入到内部数组的操作在寻路之后做了，而事实上程序语句顺序没问题。又试了几次，偶然发现，有时候有些球点击是没有反应的，也就意味着内部数组事实上数值应该是 0，也就是在随机增加小球这一块出现了问题，而且是内部数组的问题。有了大致方向找 bug，发现是自己粗心,随机赋值给小球时应该是rand()%7+1，而就在那一处没有加一，导致出现了 0 号小球。所以在找路时，被当成了“空白格”走！

解决这一个“小”bug 之后，顺带的也解决了当五个及以上小球消去时，无关的小球也被“抹掉”的奇异现象。（因为我在最后输出时是判断数组内部值是否为 0，是 0，则打印白底）

4.5 点击空白时

具体情况是第一步走完之后，走第二步时点击空白区域，就会在其他位置出现一颗小球，跑到点击的空白位置。也就是说，点击空白，直接读成了终点坐标！为了验证猜想，多试了几步，发现的确，出现小球的“其他位置”都满足一个条件-----在上一步的起点位置。所以就估计是自己在鼠标点击位置是起点还是终点的条件判断上出了问题，修改了条件后解决 bug！

4.6 依旧存在的问题----菜单返回

菜单 2 和菜单 7 的任务执行时，本该输入 End 退出，返回菜单页面，但是不知道什么原因，结果没有到菜单页面，而是显示“按任意键继续。。”然后就关闭了执行程序。代码的构成基本是一致的，同时使用的是同一个 End（）函数。

5. 心得体会

5.1 经验教训

（1 ）这次没有注意，但是在后续调试过程中深有体会，代表同一个对象的数据或者说可以相互转换的数据，尽量要去使用一致的标准，比如说都是从 0 开始或者都是从 1 开始，以及公差尽量保持一直！不然在后续调试中会花费很多心思。

（2）同上一次综合题，等号和赋值符号，这真的是一个超级隐蔽的错误！检查很费时间和眼力，就算看到了也不一定能反应过来，实在太多=和==号，麻木了

5.2 问题分解与函数重用

5.2.1 问题分解

在正式开始写程序以及老师下发具体要求之前，写过设计报告！在老师问题的引导下，基本确定了整个程序的基本思路，包括内部以什么存储方式，怎么寻路，怎么判断连线得分，将几个主要的问题想清楚之后，剩下的就是往里面加装饰，以及具体的操作过程中遇到大大小小的错误。

将一个大的问题拆分成一个个小的问题，然后再解决每个小问题的时候，再用不同的函数去解决更小的问题。写程序某种程序上来说，就是一种递归的过程。

最后的组合，我相信，熟能生巧，不是一天两天能练成的，但是平时多注意，会掌握的。

5.2.2 函数重用

在写程序之前，先看了题目要求，前三题都是打印内部数组，后三题需要打印伪图形界面。显然，一个内部数组的打印需要被用到很多次，而且每次要求都是不同的。为了方便，同时为了减少重负冗杂代码，随着函数功能的增加，逐渐增加参数来满足需求。

其实我在写程序，整理思路的时候，基本就可以把要写哪几个函数，对应要把哪些问题解决，大致可以确定下来，当然在实践过程中碰到新的问题继续改进，但基本大的函数应该不会变。只有尽可能的去满足功能的需求。

这次作业做得比较毛糙，尤其是最后两个小题，没有进行合理的函数拆分和组合，一股脑儿全堆在一起,但我想，考试过去有时间我一定会重新整理完善的！

6. 源代码(主要函数)

函数功能：实时读取鼠标所在位置的 X,Y	`void bgn_fnl_mouse(char begin[2], char final[2], int rows, int cols, int Y, int p[][10])`
	`{`
	`  HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);`
	`  const HANDLE hin = GetStdHandle(STD_INPUT_HANDLE);`
	`  int loop = 1;`
	`  int click_begin = 0;`
	`  enable_mouse(hin);`
	`  int action, x, y;`
	`  setcursor(hout, CURSOR_INVISIBLE);`
	`  while (loop)`
	`  {`
	`  action = read_mouse(hin, x, y);//读取鼠标的动作和位置`

值。

函数结束后，起点和终点的坐标已经存储在对应的数组内，为 search 提供数据。

```
            gotoxy(hout, 0, Y);
            setcolor(hout, COLOR_BLACK, COLOR_HWHITE);
            int y_i = (y - 1) / 2;//0-rows-1
            int x_j = x / 4;//0-cols-1
            if ((y - 1) / 2 + 1 > 0 && (y - 1) / 2 + 1 <= rows && (x / 4 + 1) > 0 && (x / 4 + 1) <= cols)
            {
                  cout << "[当前坐标]" << char(y_i + 'A') << "行" << x_j + 1 << "列";
                  switch (action)
                  {
                  case MOUSE_RIGHT_BUTTON_CLICK:
                        end();
                        break;
                  case MOUSE_LEFT_BUTTON_CLICK:
                        if (click_begin != -1 && p[y_i][x_j] != 0)
                        {
                              if (click_begin)
                                    showstr(hout, 2 + (begin[1] - '1') * 4, (begin[0] - 'A') * 2 + 2, "○", p[begin[0] - 'A'][begin[1] - '1'] + 1, COLOR_HWHITE);
                              begin[0] = char(y_i + 'A');
                              begin[1] = char(x_j + '1');
                              showstr(hout, 2 + x_j * 4, y_i * 2 + 2, "◎", p[y_i][x_j] + 1, COLOR_HWHITE);
                              click_begin++;
                              break;
                        }
                        else if (click_begin >0 && p[y_i][x_j] == 0)//
                        {
                              final[0] = char(y_i + 'A');
                              final[1] = char(x_j + '1');
                              loop = 0;
                              break;
                        }

                  }
            }
      }
}
```

函数功能：

伪图形界面下“小球”的移动。

```
void move_img(int p[][10], int rout[][2], char begin[2], char final[2], int rows, int cols, int Y)///如果不需要对rows,cols进行修改，用整型就可以了，不需要用指针
{
      HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
      int begin_value = p[begin[0] - 'A'][begin[1] - '1'];
      int num_pace;
      while (1)
      {
            num_pace = 0;
            for (int i = 0; i < 81; i++)
                  for (int j = 0; j < 2; j++)
                        rout[i][j] = -1;
            bgn_fnl_mouse(begin, final, rows, cols, Y, p);//给起点和终点赋值1-cols,A-ROWS
            int begin_value = p[begin[0] - 'A'][begin[1] - '1'];
            if (search(p, rout, begin[1] - '1', begin[0] - 'A', final[1] - '1', final[0] - 'A', rows, cols, &num_pace))
            {
                  reset_array(p, &cols, &rows);
                  show_rout(rout, begin_value);
                  break;
            }
            else
            {
                  reset_array(p, &cols, &rows);
```

```
                cout << "错误，无法从" << begin[0] << begin[1] << "转移到" << final[0] << final[1];
                continue;
            }
        }
}
```

函数功能：判断是否得分

```
void judge(int(*p)[10], char final[2], const int rows, const int cols, int *score, int *eve_score)
{
    int new_chess = p[final[0] - 'A'][final[1] - '1'];
    int x = final[1] - '1', y = final[0] - 'A';
    int i1 = 1, i2 = 1, i3 = 1, i4 = 1, j1 = 1, j2 = 1, j3 = 1, j4 = 1, sum = 0;

    if (x >= 1 && new_chess == p[y][x - 1] || x<cols - 1 && new_chess == p[y][x + 1])
    {
        while (x - i1 >= 0 && p[y][x - i1] == new_chess)
            i1++;
        while (x + j1<cols&&p[y][x + j1] == new_chess)
            j1++;
    }
    if (y >= 1 && p[y - 1][x] == new_chess || y < rows - 1 && p[y + 1][x] == new_chess)
    {
        while (p[y - i2][x] == new_chess && y - i2 >= 0)
            i2++;
        while (p[y + j2][x] == new_chess && y + j2<rows)
            j2++;
    }
    if (x >= 1 && y >= 1 && p[y - 1][x - 1] == new_chess || x < cols - 1 && y < rows - 1 && new_chess == p[y + 1][x + 1])
    {
        while (p[y + i3][x + i3] == new_chess && y + i3<rows && x + i3<cols)
            i3++;
        while (p[y - j3][x - j3] == new_chess &&y - j3 >= 0 && x - j3 >= 0)
            j3++;
    }
    if (x<cols - 1 && y >= 1 && p[y - 1][x + 1] == new_chess || x >= 1 && y < rows - 1 && new_chess == p[y + 1][x - 1])
    {
        while (p[y - i4][x + i4] == new_chess && y - i4 >= 0 && x + i4<cols)
            i4++;
        while (p[y + j4][x - j4] == new_chess && x - j4 >= 0 && y + j4<rows)
            j4++;
    }
    if (i1 + j1 >= 6)
    {
        sum += (i1 + j1 - 1);
        for (int i = x + 1 - i1; i < x + j1; i++)
            p[y][i] = 0;
    }
    if (i2 + j2 >= 6)
    {
        sum += (i2 + j2 - 1);
        for (int i = y - i2 + 1; i < y + j2; i++)
            p[i][x] = 0;
    }
    if (i3 + j3 >= 6)
    {
        sum += (i3 + j3 - 1);
        for (int i = 1 - j3; i < i3; i++)
            p[y + i][x + i] = 0;
    }
    if (i4 + j4 >= 6)
```

```
        {
            sum += (i4 + j4 - 1);
            for (int i = 1 - i4; i<j4; i++)
                p[y + i][x - i] = 0;
        }

        if (sum)
        {
            *eve_score = (sum - 1)*(sum - 2);
            *score += (sum - 1)*(sum - 2);
        }
        else
            *eve_score = 0;
}
```

函数功能：递归寻找路径

```
int search(int(*p)[10], int(*rout)[2], int x1, int y1, int x2, int y2, int rows, int cols,
int *num_pace)
{
    if (x1 >= 0 && x1 < cols&&y1 >= 0 && y1 < rows)
    {
        if (x1 == x2&&y1 == y2)
        {
            rout[*num_pace][0] = y1;
            rout[*num_pace][1] = x1;
            (*num_pace)++;
            return 1;
        }
        else
            p[y1][x1] = -1;
        if (p[y1][x1 + 1] == 0)
        {
            search(p, rout, x1 + 1, y1, x2, y2, rows, cols, num_pace);
            if (*num_pace)
            {
                rout[*num_pace][0] = y1;
                rout[*num_pace][1] = x1;
                (*num_pace)++;
                return 1;
            }
        }
        if (p[y1 - 1][x1] == 0)
        {
            search(p, rout, x1, y1 - 1, x2, y2, rows, cols, num_pace);
            if (*num_pace)
            {
                rout[*num_pace][0] = y1;
                rout[*num_pace][1] = x1;
                (*num_pace)++;
                return 1;
            }
        }
        if (p[y1][x1 - 1] == 0)
        {
            search(p, rout, x1 - 1, y1, x2, y2, rows, cols, num_pace);
            if (*num_pace)
            {
                rout[*num_pace][0] = y1;
                rout[*num_pace][1] = x1;
                (*num_pace)++;
                return 1;
            }
        }
        if (p[y1 + 1][x1] == 0)
```

```
            {
                search(p, rout, x1, y1 + 1, x2, y2, rows, cols, num_pace);
                if (*num_pace)
                {
                    rout[*num_pace][0] = y1;
                    rout[*num_pace][1] = x1;
                    (*num_pace)++;
                    return 1;
                }
            }
            return 0;
        }
        else
            return 0;
}
```

```
void seven(int(*p)[10])
{
    HANDLE hout = GetStdHandle(STD_OUTPUT_HANDLE);
    const HANDLE hin = GetStdHandle(STD_INPUT_HANDLE);
    int rows, cols, X, Y;
    char begin[2], final[2];
    int next[3], rout[81][2];
    init(&rows, &cols, p, 5);//初始化+随机产生球
    const int *row_ptr = &rows;//防止不小心对cols rows篡改
    const int *col_ptr = &cols;
    int score = 0, eve_score = 0;
    int leap = 0;
    print_console2(rows, cols, p);
    getxy(hout, X, Y);
    print_console3(rows, cols, next, 0);
    for (int i = 0; i < 3; i++)
        next[i] = rand() % 7;
    print_console3(rows, cols, next, 4);
    while (1)
    {
        int num_pace = 0;
        bgn_fnl_mouse(begin, final, rows, cols, X, p);//输入起点和终点的坐标1-（rows）
        int begin_value = p[begin[0] - 'A'][begin[1] - '1'];
        if (search(p, rout, begin[1] - '1', begin[0] - 'A', final[1] - '1', final[0] -
'A', *row_ptr, *col_ptr, &num_pace))
        {
            reset_array(p, col_ptr, row_ptr);//search是吧探索过的店全都变成了-1要变
回来，包括起始点
            p[final[0] - 'A'][final[1] - '1'] = begin_value;
            show_rout(rout, begin_value);//显示移动路径
            for (int i = 0; i < 81; i++)
                for (int j = 0; j < 2; j++)
                    rout[i][j] = -1;//清空存储的路径记录
            judge(p, final, *row_ptr, *col_ptr, &score, &eve_score);//判断能否得分，
若得分，设置为0，更新内部数组
            if (eve_score == 0)
            {
                update_array(p, next, col_ptr, row_ptr, 1);
                for (int i = 0; i < 3; i++)
                    next[i] = rand() % 7+1;////
                print_console3(rows, cols, next, 4);
            }
            else
            {
                for (int i = 0; i < rows; i++)
                    for (int j = 0; j < cols; j++)
                        if (p[i][j] == 0)
                            showstr(hout, 2 + j * 4, (i + 1) * 2, "  ",
COLOR_HWHITE, COLOR_HWHITE);
```

```
                    setcolor(hout, COLOR_HWHITE, COLOR_BLACK);
                    gotoxy(hout, 4 * cols + 12, 2);
                    cout << score;
                }
            }
            else
            {
                p[begin[0] - 'A'][begin[1] - '1']=begin_value;
                showstr(hout, 2 +(begin[1] - '1') * 4, (begin[0] - 'A' + 1) * 2, "○",
begin_value + 1, COLOR_HWHITE);
                reset_array(p, col_ptr, row_ptr);
                continue;
            }
            if (game_over(p, row_ptr, col_ptr,Y-1))
                break;
        }
        end();
}
```

装 订 线