同济大学 实验报告

消灭星星游戏的设计与实现

姓　　名：苗君文

学　　号：2253893

班　　级：信 06

完成日期：2023 年 6 月 13 日

二〇二三年六月

1. 题目及基本要求描述

1.1.题目描述

消灭星星，是一款消除类益智类游戏，游戏规则十分简单：玩家只需点击两个或两个以上颜色相同的星星即可消除，这样通过消除彼此相连的星星来获得积分。

本题最终要实现伪图形界面下，显示五种不同颜色的星星，并可以用鼠标点击或键盘移动来选中星星，并选中该星星周围与之同色的所有星星，通过颜色的反显来表示，而后通过鼠标点击或键盘操作来消除选中的星星们，并反复此操作，直至无法再消除，若未完全消除，则需显示剩余不可消除的星星个数，同时，本关卡结束，进入下一关继续消灭星星，过程中可以通过鼠标右键或按键盘的Q键来退出游戏。

为实现以上的最终效果，本题将分为以下多个小题来完成，并通过菜单选择的形式来呈现：

子题目A：命令行找出可消除项并标识；

子题目B：命令行完成一次消除（分步骤显示）；

子题目C：命令行完成一关（分步骤显示）；

子题目D：伪图形界面下用鼠标选择一个色块（无分隔线）；

子题目E：伪图形界面下用鼠标选择一个色块（有分隔线）；

子题目F：伪图形界面完成一次消除（分步骤）；

子题目G：伪图形界面完整版；

退出项Q：退出。

1.2.基本要求

1.2.1. 游戏规则

游戏的判断规则：输入或选中的坐标位置处不允许值为0；若输入坐标位置无相邻相同值（非0），要提示出错并重新输入；如果整个数组无相邻位置值（非0）相等，提示游戏结束；若一关结束，则游戏不结束，继续下一关，直至点击鼠标右键或按下键盘Q键结束。

分数累加规则：本次新增得分 = 消除个数^2 * 5

1.2.2. 伪图形要求

要求 cmd 窗口的大小随着输入的行列数动态变化；不同数字的前景色/背景色各不相同；画图过程中适当加延时；鼠标移动时，经过的色块用反显方式(框线用不同颜色显示) 以区分，同时下方显示经过的行列值，如果鼠标移出范围，则显示非法位置，同时当前色块的反显取消；在同一色块的行列范围内移动时，要求色块不能闪烁；有分割线时，鼠标停在色块中间的分割线时，算非法位置；如果某次消除后无法找到可消除位置，则提示本关结束；一关结束后给出奖励得分，按键后继续下一关。

2. 整体设计思路

2.1.棋盘和星星的表示

使用二维数组来表示棋盘，根据输入的行数和列数来确定棋盘的行列值，其中每个元素代表一个格子。每个格子包含星星所代表的数值，不同的星星数值代表不同的颜色。此外，可以通过用简单的运算，让不同的数值来表示不同的颜色。这样的数据结构可以更方便地访问和更新棋盘。

棋盘和星星的表示方式分为两种，分别是无分隔线和有分隔线。两种方式都可通过使用中文制表符来实现，无分隔线的情况，一个格子占3行6列；有分隔线的情况下，一个格子占4行8列。格子的尺寸对后续程序的实现十分重要，因为会根据格子大小来计算显示的位置。

2.2.星星的生成和布局

在游戏开始或每个关卡开始时，可以使用随机数种子来生成星星内部的随机数组。为形成白色棋盘为底，彩色星星置于其上的效果，应先生成白色棋盘，再通过 cct_gotoxy() 函数将光标移动到指定位置来打印星星的图案。

其中，无分割线的情况，只需打印最外层的边框，中间部分只需简单地用空格来显示即可。而有分隔线的情况，除需打印最外层的边框意外，还需打印内部的分隔线，主要通过循环结构来实现。而后在指定位置打印星星，两种情况可以共用一个打印星星的函数。而位置的区别则在于格子的大小：3*6的格子与4*8的格子，此处需要注意数学计算。

2.3.选择星星

本题使用鼠标移动并用左键点击的方式/键盘操作的方式来选择星星。最重要的是学习 test-cct.cpp 中“鼠标键盘操作演示”的部分，可以根据它的选择结构来改写。鼠标滑动时所经过的星星需要用反显方式（框线用不同颜色显示）来与其他星星以示区别，意味着打印星星的函数需要用一个参数来选择不同的前景颜色（黑/亮白）。

如果没有点击鼠标左键，或按下键盘的回车键选中这个星星，若未离开这个星星，则不需要有变动；若离开了这个星星，则这个星星的颜色应变回正常的以黑色为前景颜色的状态。在没有离开这个星星的时候，星星需要保持现有的颜色状态，也就可以确保在同一色块的行列范围内移动时，色块不会闪烁。

那么关键在于通过光标位置来计算坐标所在的星星的行列数。另外，在有分隔线的情况下，要关注当光标位置在分隔线上时，属于非法位置。

2.4.星星消除

实现星星消除的逻辑是游戏的核心，可以采用递归或者迭代的方式遍历棋盘，检查是否存在连续的相同颜色的星星，并标记待消除的星星。为了标记星星，需要另外设置一个用来标记的数组，未标记的星星，数组元素为0；被标记的星星，数组元素为1。然后执行消除操作，星星消失，即星星所代表的值

变为0。

而后需要下移星星，以补全被消除星星的位置。此处，对于内部数组，为了下移数组元素，可以使用冒泡排序法来实现数组下落的操作。对于伪图形界面的显示，可以设置一个星星下落的函数，根据数组下落的函数增加图形的显示。值得注意的是，需要以每一个cmd小格子来移动，并增加延时效果，从而形成动画效果。

下移星星的操作包括从上至下，从右至左。

2.5.计分规则

每一次成功消除星星后，计算本次得分，并计算总得分，并在（0,0）处打印分数。每一次的计分根据当次消除的星星数来计算，也就意味着需要一个变量来计数，可以通过逐个查找数组中被标记的元素，来获得当次消除星星的个数，此处可以用一个计算本次得分的函数来计算。

当一个关卡结束时，进入奖励分数的函数计算奖励分，奖励分根据无法消除的星星的个数来计算，因此还需传参。计算后，再进入下一关，同时分数继续累加，不置为零。

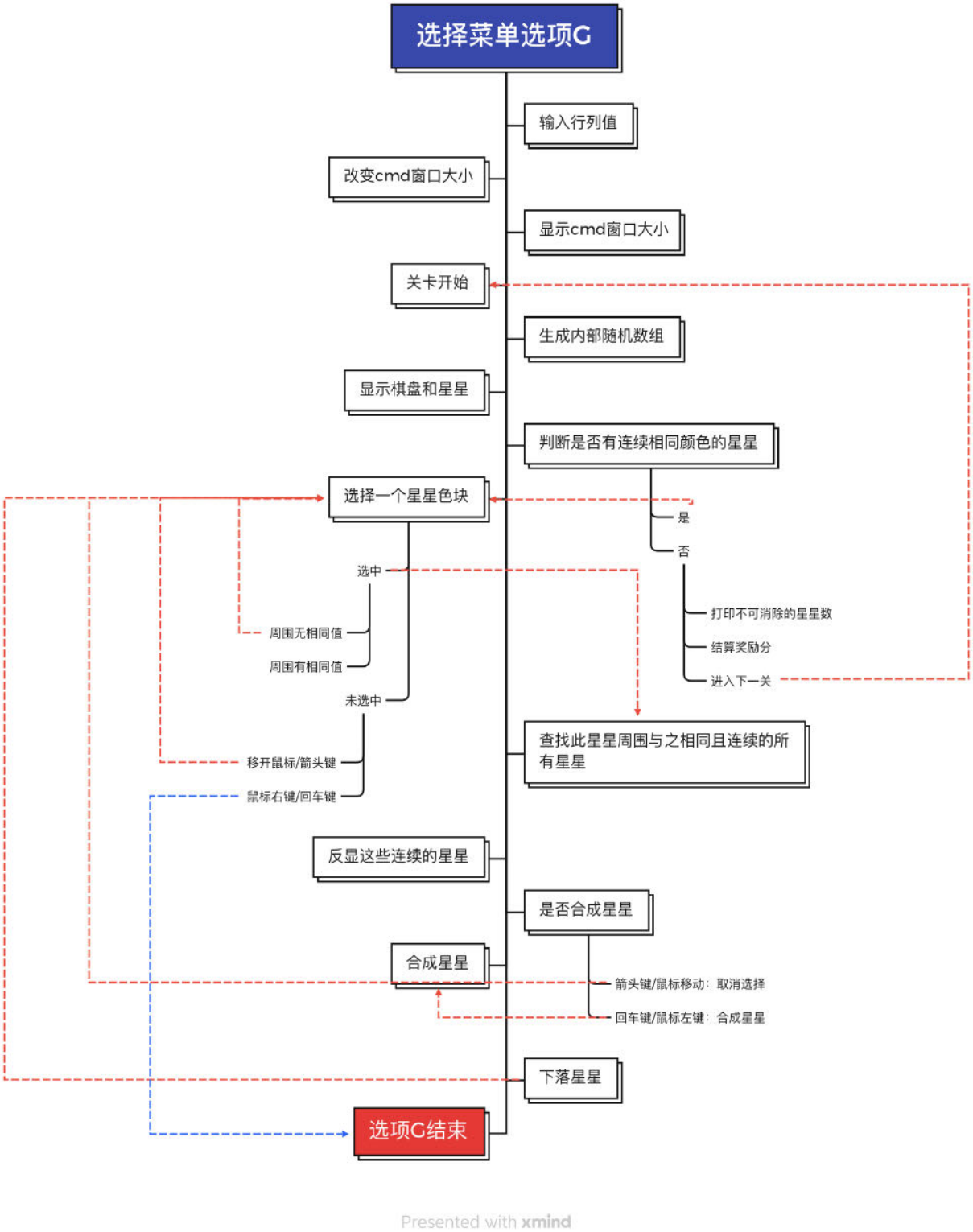
2.6.结束条件和胜利条件

本题中结束子题目的条件为：点击鼠标右键或点击键盘上的Q键来退出。此功能可以根据 test-cct.cpp 中“鼠标键盘操作演示”这一部分来仿写。

本题中结束当前关卡的条件为：不再有可以消除的星星，并显示无法消除星星的个数。此处需要有一个可以判断有无可消除项的函数来判断。结束当前关卡后进入新的关卡，开始新的游戏，也就是说需要一个循环，来保证关卡的循环。

本题中的胜利条件为：所有星星都被消除。可以用一个判断胜利的函数来决定，逐个检查是否每个数组元素都为0.

3. 主要功能的实现


选择菜单选项G
输入行列值
改变cmd窗口大小
显示cmd窗口大小
关卡开始
生成内部随机数组
显示棋盘和星星
判断是否有连续相同颜色的星星
是
否
打印不可消除的星星数
结算奖励分
进入下一关
选择一个星星色块
选中
周围无相同值
周围有相同值
未选中
移开鼠标/箭头键
鼠标右键/回车键
查找此星星周围与之相同且连续的所有星星
反显这些连续的星星
是否合成星星
合成星星
箭头键/鼠标移动：取消选择
回车键/鼠标左键：合成星星
下落星星
选项G结束
Presented with xmind

4. 调试过程碰到的问题

4.1.在单击一次左键后，鼠标后续无法进行单击左键的操作

解决方法：在发现点击鼠标左键后无法执行相应程序后，我首先仔细察看了一遍我的源程序，认为执行操作基本没有问题，因此我认为主要错误出在cct函数的使用上。

于是，我仔细观察了test-cct.cpp文件中“鼠标键盘操作演示”的部分，我发现给出的程序在调用与鼠标或键盘相关的函数时，都在调用前有一句cct_enable_mouse，在调用结束后有一句cct_disable_mouse，而我原本的程序中并没有在每一次判断鼠标或键盘的操作的前后调用cct_enable_mouse和cct_disable_mouse这两个函数。

为了弄清楚是不是这两个函数的调用而导致的问题，我查看了cmd_console_tools.cpp这个文件中关于以上两个函数的代码和注释。我发现cct_enable_mouse的功能是允许使用鼠标，而某些cmd窗口控制语句执行后，可能会取消鼠标支持，则需调用该函数回再次加入。类似的，cct_disable_mouse的功能是禁用鼠标，允许使用鼠标后需要重新调用该函数。

通过了解这两个函数，我意识到问题所在，并在与鼠标键盘操作相关的代码前后增加了允许使用鼠标函数和禁用鼠标函数，便可以正常用鼠标左键来操作。

4.2.循环选择星星时，总是会先在第A行第0列位置的星星闪烁

解决方法：由于是在每次选择星星时，第A行第0列位置的星星反显，直至鼠标移动或者键盘操作时才恢复正常颜色，说明问题出在选择星星这个函数上。

经过观察，我发现，我原本所写的选择星星的函数，在每次进入该函数时，会将所在的行和列先初始化在第A行第0列的位置，这是由于在前几道子题目中，并没有要求选择多次星星，都是只操作一次，而第一次选择时，会先初始化星星的位置：第A行第0列。但多次选择星星时，第二次及以后的选择都不需要再初始化星星的位置了。

也就是说，需要一个参数来决定这一次选择星星需不需要初始化位置。通过观察demo可知，进入新的关卡后，第一次需要初始化选择星星的位置，其余时刻都不需要初始化。

因此我增加了选择星星的函数的传参数，来控制是否选择初始化位置。而在实现子题目G的主过程中，增加了参数int initial_0,在每次关卡开始时，将initial_0置为1；在调用过一次选择星星的函数后，将initial_0置为0，从而就可以解决该问题了。

5. 心得体会

5.1.完成本次作业得到的心得体会、经验教训

通过本次作业，我认为在完成一个项目时，首先要想好思路。针对本题，思路可以从子题目的设计上来汲取。内部的逻辑是根据内部数组来变化；而伪图形界面则需要根据内部数组，借助cct工具来写。另外，函数的合理划分也尤其重要。

在子题目A,B,C中我有将每一个函数划分的很细致，也就方便调用，三者之间不同的内容则是输入输出部分以及循环与否的问题。将每一步调整数值数组与标记数组的变化都放到不同的数组中，当程序出错时，也可以方便快速地知道问题出在哪个函数，从而去调整函数的内容来debug。

在子题目D,E,F,G中，我由于写的时候思路不是很清晰，就将有分隔线和无分隔线分为两种函数来处理，每一步的操作都分为_noline和_line两种函数来操作，这样写在写子题目D,E时还算方便，但是在写子题目F,G时，由于循环操作导致之前并未出现的问题全都显露出来，而在debug时需要改两倍的函数，十分繁琐。

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?

我认为在完成上一次汉诺塔作业时，由于前期完成过小作业，在讲评的基础上进行debug就很方便地分解函数并加以使用，但是这一次的消灭星星作业没有前期的基础，就不是很好分解函数，导致越写越混乱。

对于前面小题已完成的代码，可以通过增加传参来针对不同情况下调用/选择函数中的不同部分来运行。为了更好地重用代码，要对整个题目的大致思路在写前有比较清晰的想法，了解前后几道小题的关联关系，在写函数时注意函数参数的设置，根据不同的函数参数，编写针对不同情况的代码，从而方便重用。

6. 附件：源程序

```
if (x == 'G') {
                int i, j, num = 0;
                int column = 0, row = 0;
                bool judge=1, judge1=1;
                int initial = 1;
                int initial_0 = 1;
                graph_line(array, mark, columns, rows);
                while (1) {
                      if (initial == 0) {
                            initial_array(array, mark,
rows, columns);
                            graph_line(array, mark,
columns, rows);
                            initial_0 = 1;
                      }
                      for (i = 0;i < rows;i++)
                            for (j = 0;j < columns;j++)
                                  mark[i][j] = 0;
                      for (i = 0;i < rows;i++)
                            for (j = 0;j < columns;j++)
                                  if (around(i, j, columns,
rows, array))
                                        judge = 1;
                      if (judge == 0) {
                            for (i = 0;i < rows;i++)
                                  for (j = 0;j <
columns;j++)
                                        if (array[i][j] !=
0)
                                              num++;
                            cct_setcolor();
                            cct_gotoxy(0, 3 + 4 * rows);
                            cout << "
";
                            cct_gotoxy(0, 3 + 4 * rows);
                            cct_setcolor(COLOR_HYELLOW,
COLOR_RED);
                            cout << "剩余" << num << "个星
星，无可消除项，本关结束!";
                            cct_setcolor();
                            cct_gotoxy(0, 0);
                            cout << "
";
                            cct_gotoxy(0, 0);
                            cout << "奖励得分：" <<
reward_score(num);
                            score += reward_score(num);
                            cout << "  总得分：" << score;
                            enter_next();
                            initial = 0;
                            continue;
                      }
                      while (1) {
                            if (choose_star_line(column,
row, columns, rows, array, mark, initial_0) == 0) {
                                  judge1 = 0;
                                  break;
                            }
                            if (around(row, column,
columns, rows, array) == 0) {
                                  judge1 = 1;
                                  cct_setcolor();
                                  cct_gotoxy(0, 3 + 4 *
rows);
                                  cout << "
";
                                  cct_gotoxy(0, 3 + 4 *
rows);
                                  cct_setcolor(COLOR_BLACK,
COLOR_HYELLOW);
                                  cout << "周围无相同值，请
重新选择";
                                  cct_setcolor();
                                  cct_enable_mouse();

      cct_setcursor(CURSOR_INVISIBLE);
                                  mark[row][column] = 0;
                                  while (1) {
                                        int X = 0, Y = 0;
                                        int ret, maction;
                                        int keycode1,
keycode2;
                                        ret =
cct_read_keyboard_and_mouse(X, Y, maction, keycode1,
keycode2);
                                        if (ret ==
CCT_MOUSE_EVENT) {
                                              if (X >= 10 +
8 * column || X <= 3 + 8 * column || Y >= 6 + 4 * row || Y
<= 2 + 4 * row) {
                                                    break;
                                              }
                                        }
                                        else if (ret ==
CCT_KEYBOARD_EVENT) {
                                              break;
                                        }
                                  }
                                  cct_disable_mouse();
                                  paint_star(4 + 8 *
column, 3 + 4 * row, array[row][column], 0);
                                  initial = 0;
                                  initial_0 = 0;
                                  continue;
                            }
                            else {
                                  check_same(array, mark,
column, row, columns, rows);
                                  for (i = 0;i < rows;i++)
                                        for (j = 0;j <
columns;j++)
                                              if (mark[i][j]
== 1)

      paint_star(4 + 8 * j, 3 + 4 * i, array[i][j], 1);
                                  if
(star_lump_line(column, row, columns, rows, array, mark) ==
1) {
                                        initial = 0;
                                        continue;
                                  }
                                  stars_white_line(columns,
rows, column, row, array, mark, score);
                                  cct_setcolor();
                                  stars_fall_line(array,
mark, columns, rows);
```

```
                              initial = 1;
                              initial_0 = 0;
                              break;
                        }
                  }
                  if (judge1 == 0)
                        break;
            }
            cct_gotoxy(0, 5 + 4 * rows);
            wait_for_end();
      }
void graph_line(int array[][10], int mark[][10], int columns,
int rows)
{//画有边框的白底
      cct_setconsoleborder(5 + 8 * columns, 7 + 4 * rows);
      cct_gotoxy(0, 0);
      cct_setcolor();
      cout << "屏幕当前设置为：" << 7 + 4 * rows << "行" <<
5 + 8 * columns << "列" << endl;
      for (int i = 0;i < columns;i++) {
            cct_gotoxy(6 + 8 * i, 1);
            cout << i;
      }
      cct_gotoxy(2, 2);
      cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
      cout << "╔";
      for (int i = 0;i < columns-1;i++)
            cout << "═══════╦";
      cout << "═══════╗";
      for (int i = 0;i < rows-1;i++) {
            for (int j = 0;j < 3;j++) {
                  if (j == 1) {
                        cct_setcolor();
                        cct_gotoxy(0, 3 + 4 * i + j);
                        cout << char('A' + i);
                  }
                  cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
                  cct_gotoxy(2, 3 + 4 * i + j);
                  cout << "║";
                  for (int k = 0;k < columns;k++) {
                        cout << "       "<< "║";
                        Sleep(1);
                  }
            }
            cct_gotoxy(2, 3 + 4 * i + 3);
            cout << "╠";
            for (int k = 0;k < columns - 1;k++)
                  cout << "═══════╬";
            cout << "═══════╣";
      }
      for (int i = 0;i < 3;i++) {
            cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
            cct_gotoxy(2, 4 * rows + i - 1);
            cout << "║";
            for (int j = 0;j < columns;j++) {
                  cout << "       ║";
                  Sleep(1);
            }
      }
      cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
      cct_gotoxy(2, 2 + 4 * rows);
      cout << "╚";
      for (int i = 0;i < columns-1;i++)
            cout << "═══════╩";
      cout << "═══════╝";
      for (int i = 0;i < rows;i++) {
            for (int j = 0;j < columns;j++) {
                  paint_star(4 + 8 * j, 3 + 4 * i, array[i][j],
0);
            }
      }
}
void paint_star(int x, int y, int array, bool choose)
{
      int bg_color, fg_color;
      bg_color = array + 9;
      if (choose)
            fg_color = COLOR_HWHITE;
      else
            fg_color = COLOR_BLACK;
      if (array == 0) {
            fg_color = COLOR_HWHITE;
            bg_color = COLOR_HWHITE;
      }
      cct_gotoxy(x, y);
      cct_setcolor(bg_color, fg_color);
      cout << "      ";
      cct_gotoxy(x, y + 1);
      cout << "  ★  ";
      cct_gotoxy(x, y + 2);
      cout << "      ";
}
void star_position_line(int action, int& column, int& row,
int columns, int rows, int array[][10], int mark[][10], int
x, int y)
{//有边框，输出星星位置
      int tc = -1, tr = -1, position = 1;
      x = x - 4;
      y = y - 3;
      if (x < 0 || y < 0 || x >= 8 * columns || y >= 4 * rows
|| x % 8 == 6 || x % 8 == 7 || y % 4 == 3) {
            tc = -1;
            tr = -1;
      }
      else {
            tc = x / 8;
            tr = y / 4;
      }
      if (column == tc && row == tr && action == 1)
            return;
      if (x < 0 || y < 0 || x >= 8 * columns || y >= 4 * rows
|| x % 8 == 6 || x % 8 == 7 || y % 4 == 3)
            position = 0;
      else if (mark[x][y] == 0)
            position = 0;
      cct_setcolor();
      cct_gotoxy(11, 3 + 4 * rows);
      if (position == 0)
            cout << "位置非法";
      else
            cout << char('A' + tr) << "行" << tc << "列     "
<< endl;
      return;
}
bool choose_star_line(int& column, int& row, int columns,
int rows, int array[][10], int mark[][10], int initial)
{
      //有边框，选中星星
      int X = 0, Y = 0;
      int ret, maction;
      int keycode1, keycode2;
      int loop = 1;
      int tc = 0, tr = 0;
      int keyboard = 1, mouse = 1;
```

```
	bool judge = 1;
	cct_enable_mouse();
	cct_setcursor(CURSOR_INVISIBLE);
	if (initial == 1) {
		column = 0;
		row = 0;
		cct_setcolor();
		cct_gotoxy(0, 3 + 4 * rows);
		cout << "箭头键/鼠标移动，回车键/单击左键选择并结束，Q/单击右键结束";
	}
	if (array[row][column] != 0)
		paint_star(4 + 8 * column, 3 + 4 * row, array[row][column], 1);
	while (loop) {
		ret = cct_read_keyboard_and_mouse(X, Y, maction, keycode1, keycode2);
		if (column != -1 && row != -1) {
			tc = column;
			tr = row;
		}
		if (ret == CCT_MOUSE_EVENT) {
			if (maction == MOUSE_ONLY_MOVED) {
				if (mouse == 1) {
					cct_setcolor();
					cct_gotoxy(0, 3 + 4 * rows);
					cout << "
";
					cct_gotoxy(0, 3 + 4 * rows);
					cout << "[当前鼠标] ";
					mouse = 0;
					keyboard = 1;
				}
				star_position_line(1, column, row, columns, rows, array, mark, X, Y);
				star_change_line(1, column, row, columns, rows, array, mark, X, Y);
				continue;
			}
			if (maction == MOUSE_LEFT_BUTTON_CLICK) {
				star_position_line(1, column, row, columns, rows, array, mark, X, Y);
				star_change_line(1, column, row, columns, rows, array, mark, X, Y);
				if (column == -1 && row == -1)
					continue;
				cct_setcolor();
				cct_gotoxy(0, 3 + 4 * rows);
				cout << "
";
				cct_gotoxy(0, 3 + 4 * rows);
				cout << "选中了" << char('A' + row) << "行" << column << "列 " << endl;
				cct_disable_mouse();
				judge = 1;
				break;
			}
			if (maction == MOUSE_RIGHT_BUTTON_CLICK) {
				judge = 0;
				cct_disable_mouse();
				break;
			}
		}
		else if (ret == CCT_KEYBOARD_EVENT) {
			if (row == -1 && column == -1) {
				column = tc;
				row = tr;
			}
			if (keycode1 == 224) {
				if (keyboard == 1) {
					cct_setcolor();
					cct_gotoxy(0, 3 + 4 * rows);
					cout << "
";
					cct_gotoxy(0, 3 + 4 * rows);
					cout << "[当前键盘] ";
					keyboard = 0;
					mouse = 1;
				}
				tc = column;
				tr = row;
				switch (keycode2) {
					case KB_ARROW_UP:
						row = (row - 1 + rows) % rows;
						break;
					case KB_ARROW_DOWN:
						row = (row + 1) % rows;
						break;
					case KB_ARROW_LEFT:
						column = (column - 1 + columns) % columns;
						break;
					case KB_ARROW_RIGHT:
						column = (column + 1) % columns;
						break;
				}
				star_position_line(0, tc, tr, columns, rows, array, mark, 4 + 8 * column, 3 + 4 * row);
				star_change_line(0, tc, tr, columns, rows, array, mark, 4 + 8 * column, 3 + 4 * row);
				continue;
			}
			if (keycode1 == 13) {
				cct_setcolor();
				cct_gotoxy(0, 3 + 4 * rows);
				cout << "
";
				cct_gotoxy(0, 3 + 4 * rows);
				cout << "选中了" << char('A' + row) << "行" << column << "列" << endl;
				cct_disable_mouse();
				judge = 1;
				break;
			}
			if (keycode1 == 'q' || keycode1 == 'Q') {
				judge = 0;
				cct_disable_mouse();
				break;
			}
		}
	}
	cct_disable_mouse();  //禁用鼠标
	cct_setcursor(CURSOR_VISIBLE_NORMAL);  //打开光标
	return judge;
}
bool star_lump_line(int& column, int& row, int columns, int rows, int array[][10], int mark[][10])
{
	int i, j;
	cct_setcolor();
	cct_gotoxy(0, 3 + 4 * rows);
```

```
	cout                              <<                              "
";
	cct_gotoxy(0, 3 + 4 * rows);
	cout << "箭头键/鼠标移动取消当前选择，回车键/单击左键合成";
	bool judge = 0;//judge 为 1 意味着重新选择，需要重新合成
	cct_enable_mouse();
	while (1) {
		int X = 0, Y = 0;
		int ret, maction;
		int keycode1, keycode2;
		ret = cct_read_keyboard_and_mouse(X, Y, maction, keycode1, keycode2);
		cct_enable_mouse();
		if (ret == CCT_MOUSE_EVENT) {
			if (maction == MOUSE_ONLY_MOVED) {
				if (X >= 10 + 8 * column || X <= 3 + 8 * column || Y >= 6 + 4 * row || Y <= 2 + 4 * row) {
					judge = 1;
					break;
				}
				else
					continue;
			}
			if (maction == MOUSE_LEFT_BUTTON_CLICK)
				break;
		}
		if (ret == CCT_KEYBOARD_EVENT) {
			if (keycode1 == 224) {
				switch (keycode2) {略}
				judge = 1;
				break;
			}
			if (keycode1 == 13)
				break;
		}
	}
	cct_disable_mouse();
	if (judge) {
		for (i = 0;i < rows;i++)
			for (j = 0;j < columns;j++)
				if (mark[i][j] == 1) {
					paint_star(4 + 8 * j, 3 + 4 * i, array[i][j], 0);
					mark[i][j] = 0;
				}
	}
	return judge;
}
void stars_white_line(int columns, int rows, int column, int row, int array[][10], int mark[][10], int& score)
{
	int i, j;
	cct_setcursor(CURSOR_INVISIBLE);
	lump_same(array, mark, columns, rows);
	cct_setcolor();
	cct_gotoxy(0, 0);
	cout << "                                                  ";
	cct_gotoxy(0, 0);
	cout << "本次得分：" << this_score(array, mark, columns, rows);
	score += this_score(array, mark, columns, rows);
	cout << "  总得分：" << score;
	for (i = 0;i < rows;i++)
		for (j = 0;j < columns;j++)
			if (mark[i][j] == 1) {
				paint_star(4 + 8 * j, 3 + 4 * i, array[i][j], 0);
				Sleep(1);
			}
	mark[i][j] = 0;
}
void stars_fall_line(int array[][10], int mark[][10], int columns, int rows)
{
	int i, j, k;
	int a, m;
	bool judge = 0;
	cct_setcursor(CURSOR_INVISIBLE);
	for (i = 0;i < columns;i++)
		for (j = 0;j < rows - 1;j++) {
			for (k = 0;k < rows - 1 - j;k++) {
				if (mark[k + 1][i] == 1) {
					a = array[k][i];
					m = mark[k][i];
					array[k][i] = array[k + 1][i];
					mark[k][i] = mark[k + 1][i];
					array[k + 1][i] = a;
					mark[k + 1][i] = m;
				}
			}
			for (k = 0;k < rows - j;k++) {
				if (mark[k][i] == 1&&array[k][i]==0) {
					judge = 1;
					break;
				}
			}
			if (judge) {
				for (k = 0;k < rows - j;k++) {
					for (int t = 0;t < 3;t++) {
						cct_showch(4 + 8 * i, 3 + 4 * k + t, ' ', COLOR_HWHITE, COLOR_HWHITE, 6);
						paint_star(4 + 8 * i, 3 + 4 * k, array[k][i], 0);
						Sleep(1);
					}
					cct_gotoxy(4 + 8 * i, 3 + 4 * k + 3);
					cct_setcolor(COLOR_HWHITE, COLOR_BLACK);
					cout << "═══════";
				}
			}
			judge = 0;
		}
	for (i = columns;i > 0;i--)
		for (j = 0;j < i;j++) {
			if (array[rows - 1][j] == 0) {
				for (k = 0;k < rows;k++) {
					if (array[k][j + 1] == 0)
						continue;
					a = array[k][j];
					m = mark[k][j];
					array[k][j] = array[k][j + 1];
					mark[k][j] = mark[k][j + 1];
					array[k][j + 1] = a;
					mark[k][j + 1] = m;
					for (int t = 0; t < 9;t++) {
						cct_showch(10 + 8 * j - t, 3 + 4 * k, ' ', COLOR_HWHITE, COLOR_HWHITE, 1);
						cct_showch(10 + 8 * j - t, 3 + 4 * k + 1, ' ', COLOR_HWHITE, COLOR_HWHITE, 1);
						cct_showch(10 + 8 * j - t,
```

```
3 + 4 * k + 2, ' ', COLOR_HWHITE, COLOR_HWHITE, 1);
                                paint_star(4 + 8 * (j) - t,
3 + 4 * k, array[k][j], 0);
                                Sleep(1);
                           }
                           cct_setcolor(COLOR_HWHITE,
COLOR_BLACK);
                           cct_gotoxy(10 + 8 * j, 3 + 4 *
k);
                           cout << "┃";
                           cct_gotoxy(10 + 8 * j, 4 + 4 *
k);
                           cout << "┃";
                           cct_gotoxy(10 + 8 * j, 5 + 4 *
k);
                           cout << "┃";
                      }
                 }
           }
}
void initial_array(int array[][10],int mark[][10], int row,
int column)
{
    int i, j;
    for (i = 0;i < row;i++) {
        for (j = 0;j < column;j++)
            array[i][j] = rand() % 5 + 1;
    }
}
void initial_mark(int mark[][10], int columns, int rows)
{//将 mark 数组全初始化为 0
     int i, j;
     for (i = 0;i < rows;i++)
          for (j = 0;j < columns;j++)
               mark[i][j] = 0;
}

bool around(int x, int y, int columns, int rows, int
array[][10])//x 为 row,y 为 column
{//判断 array 周围有无相同值(非 0)
     bool judge = 0;
     if (array[x][y] == 0)
          judge = 0;
     else {
          if (x - 1 >= 0 && array[x - 1][y] == array[x][y])
               judge = 1;
          else if (x + 1 < rows && array[x + 1][y] ==
array[x][y])
               judge = 1;
          else if (y - 1 >= 0 && array[x][y - 1] ==
array[x][y])
               judge = 1;
          else if (y + 1 < columns && array[x][y + 1] ==
array[x][y])
               judge = 1;
          else
               judge = 0;
     }
     return judge;
}
void check_same(int array[][10], int mark[][10], int column,
int row, int columns, int rows)
{//逐个检查，找到周围的相同值
     int i = 0, j = 0;
     mark[row][column] = 1;
     int a = 1;
     while (a != 0) {
          a = 0;
          for (i = 0;i < rows;i++) {
               for (j = 0;j < columns;j++) {
                    if (array[i][j] == array[row][column]
&& nearself(j, i, columns, rows, mark) && mark[i][j] == 0) {
                         mark[i][j] = 1;
                         a = 1;
                    }
               }
          }
     }
}
void find_same(int array[][10], int mark[][10], int column,
int row, int columns, int rows, int t)
{//使用递归查找周围所有相同值
     if (row < 0 || row >= rows || column < 0 || column >=
columns)
          return;
     if (mark[row][column]  == 1 || array[row][column] != t)
          return;
     else {
          mark[row][column] = 1;
          find_same(array, mark, column - 1, row, columns,
rows, t);
          find_same(array, mark, column + 1, row, columns,
rows, t);
          find_same(array, mark, column, row - 1, columns,
rows, t);
          find_same(array, mark, column, row + 1, columns,
rows, t);
     }
     return;
}
Void lump_same(int array[][10], int mark[][10],int
columns,int rows)
{//归并相同值
     int i, j;
     for (i = 0;i < rows;i++)
          for (j = 0;j < columns;j++)
               if (mark[i][j] == 1)
                    array[i][j] = 0;
}
int this_score(int array[][10], int mark[][10], int columns,
int rows)
{//计算本次得分
     int score = 0, num = 0;
     int i, j;
     for (i = 0;i < rows;i++)
          for (j = 0;j < columns;j++)
               if (mark[i][j] == 1 && array[i][j] == 0)
                    num++;
     score = num * num * 5;
     return score;
}
int reward_score(int num)
{
     int reward;
     if (num >= 10)
          reward = 0;
     else if (num < 10)
          reward = 180 * (10 - num);
     return reward;
}
```