高级程序语言设计上机实验报告II

——合成十游戏

2151140 信07 王谦

2021年12月30日

1. 合成十题目描述

1.1. 题目整体目标

合成十游戏基本规则：

1. 在设定的M×N的区域内，初始产生随机值1-3。
2. 然后选中其中的一个坐标为目标，找出所有上下左右与之相邻且数值相同的坐标，对新找到的坐标继续找相邻且相同的坐标。
3. 接着将所有的相邻值合并在选中的目标位置，该位置的数值+1，被合并的其余数值归零。
4. 然后整个区域所有的值跳过0自然下落，使0留在区域上方。
5. 接下来在所有0位置随机覆盖新值1-N，其中N为当前已有数的最大值，对各覆盖值的概率另有额外规定。
6. 重复除了1初始以外的以上步骤，直到新值达到一定的目标大小。
7. 完成目标大小后，出现提示继续向更高目标进发。
8. 如果途中所有数字均无法找到相邻值合并，则游戏失败。

覆盖值的概率规定：

条件	随机数值范围	分布概率
1、初始 2、合并后最大值为 3	1-3	等概率
3、合并后最大值为 4	1-4	1~3：各 30% 4 ：10%
4、合并后最大值为 5	1-5	1~3：各 25% 4 ：15% 5 ：10%
5、合并后最大值为 6	1-6	1~4：各 20% 5 ：15% 6 ：5%
6、合并后最大值为 x (x>6)	1-x	1~x-3：80%/(x-3) 均分后取整 x-2 ：10% x-1 ：5% x ：5%

记分规则：本次新增得分=消除值×消除个数×3

本次题目便是在上述规则下，做出实现合成十游戏目标效果的程序。依照要求将题目分为多个小题以菜单形式整合。这些小题可分为命令行类、伪图形界面类和网络版等不同实现形式。具体各小题目标见下。

1.2. 菜单各项目标

1. 命令行找出可合成项并标识（非递归）
2. 命令行找出可合成项并标识（递归）

3.命令行完成一次合成（分步骤显示）
4.命令行完整版（分步骤显示）
5.伪图形界面显示初始数组（无分隔线）
6.伪图形界面显示初始数组（有分隔线）
7.伪图形界面下用箭头键 / 鼠标选择当前色块
8.伪图形界面完成一次合成（分步骤）
9.伪图形界面完整版(支持鼠标)
A.命令行界面（网络自动解 - 基本版）
B.伪图形界面（网络自动解 - 基本版）
C.命令行界面（网络自动解 - 竞赛版）
0.退出
其中0-9各目标效果以给出的demo为参考。（ABC为网络版，我没有实现）

1.3. 额外的要求与限制

常规的规范限制：如不允许使用后续知识点；不允许使用goto、C++的string类等知识内容以及相关的文件命名与提交要求规范。

全局变量的限制：本次不允许使用任何形式的全局变量、全局数组、全局指针，但是允许使用全局的宏定义和常变量。

菜单项的额外要求：未完成AB则菜单项中要去除对应项，否则扣分。

1.4.其它

字体字号、颜色、时长、提示信息具体内容、全半角、空格数等无强制要求。

可以使用之前提供的函数工具包，保证了具备足够的工具实现目标效果。

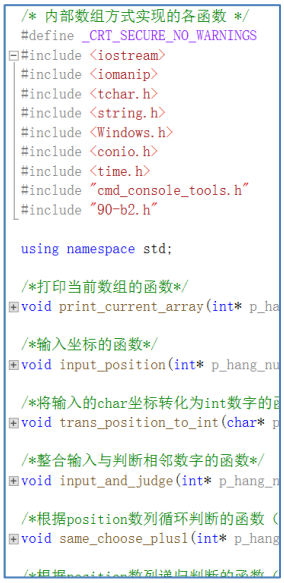
网络版提供了额外的工具及指导。

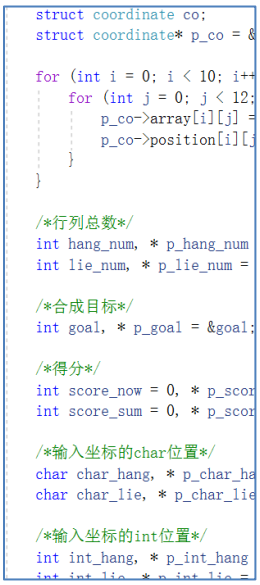
2. 整体设计思路

2.1.大致策略

将每一个问题分为许多小函数，在项目中额外建立多个cpp文件用来存放我所构造的函数，分为共用的函数、命令行的函数、伪图形界面的函数等等。在头文件中声明以达到互通的效果。在main函数中按顺序调用相应函数实现目标效果。

在main函数中设出所需要使用的各种变量并相应初始化，接着在一个循环结构中使用menu函数执行菜单选择，根据选择结果调用选择分支进入各个菜单项的执行，对应菜单项退出后重新回到循环，如此往复，如果选择0便跳出循环结束main函数。





各个菜单项由小函数组合而来。
部分举例见右图。

```
switch (menu_choice) {
    case '1':
        input_hanglie(p_hang_num, p_lie_num);
        generate_array(p_hang_num, p_lie_num, p_co);
        print_current_array(p_hang_num, p_lie_num, p_co);
        input_and_judge(p_hang_num, p_lie_num, p_char_hang, p_char_lie, p_int_
        same_choose_plus1(p_hang_num, p_lie_num, p_int_hang, p_int_lie, p_obje
        print_choose(p_hang_num, p_lie_num, p_int_hang, p_int_lie, p_object, p
        print_array_with_color(p_hang_num, p_lie_num, p_int_hang, p_int_lie, p
        need_end();
        break;
```

3. 主要功能的实现

1. 标记数组特定位置的方法：

```
struct coordinate {
    int array[10][12];
    char position[10][12];
};
```

设置一个结构体如右图，其包含两个平行的数组其中array既是我们所见的常规数组，而position则是一个隐藏着的数组，初始化全部为0，对特定的坐标就可以将position中的0替换为*，通过查找为*的position坐标就可以确定对应的array坐标。

```
struct coordinate co;
struct coordinate* p_co = &co;

for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 12; j++) {
        p_co->array[i][j] = 0;
        p_co->position[i][j] = '0';
    }
}
```

2. 递归函数确定合并项的实现过程：

若该坐标数值与目标相同并且该处没有被标记，则标记此处。
若该坐标四周数值与目标相同并且没有被标记，则标记对应四周坐标，
并在各个四周坐标再次执行此递归函数。

3. 菜单项3命令行完成一次合成（分步骤显示）的实现过程：

输入行列的函数->输入目标的函数->随机生成初始化数组
->打印当前数组->输入命令行->判断周围是否可合并，不可合并则重新输入
->循环或递归寻找所有可合并值->打印查找结果（符号表示）->打印当前数组
->提示是否合并->是则合并打印->结束函数

4. 菜单项4命令行完整版（分步骤显示）的实现过程：

输入行列的函数->输入目标的函数->随机生成初始化数组
->进入4专用的整合函数->结束函数

整合函数：
打印当前数组->输入命令行->判断周围是否可合并，不可合并则重新输入
->循环或递归寻找所有可合并值->打印查找结果（符号表示）->打印当前数组

->提示是否合并，得到返回值
->进入循环
{若Y，合并+打印+打分+下落+打印+增补+打印+回车继续+判断是否能继续合并+判断是否达到目标继续前进
若N，position全部归零
若Q，退出循环}

5. 对应窗口坐标与数组行列的确定：

举例如右图，其中i、j为行列，x、y为窗口坐标。

```
/*改变一块颜色的函数*/
void changcolor_for_one(int i, int j, coordinate* p_co, int fg)
{
    int x = j * 8 - 4;
    int y = i * 4 - 1;
    int bg = (p_co->array[i][j] % 14);
    cct_gotoxy(x, y);
    cct_setcolor(bg, fg);
    cout << "┌──┐";
    cct_gotoxy(x, y + 1);
    cout << "│" << setw(2) << p_co->array[i][j] << "│";
    cct_gotoxy(x, y + 2);
    cout << "└──┘";
    cct_setcolor();
}
```

6. 菜单项7伪图形界面下用箭头键 / 鼠标选择当前色块的实现过程：

输入行列的函数->随机生成初始化数组
->设置窗口和缓冲区大小->动画打印数组->进入键盘鼠标控制函数->结束函数

键盘鼠标控制函数：
进入循环
{position归零->读取键鼠->判断状态执行相应过程}
->判断菜单项为7->结束

```
/*下落一格动画的函数*/
void fall_for_one(int i, int j, coordinate* p_co) { ... }

/*下落一列动画的函数*/
void fall_for_one_lie(int* p_hang_num, int* p_lie_num, int i
```

```
/*下落动画除0的函数*/
void fall_visualized(int* p_hang_num, int* p_lie_num, int* p_int_hang
{
    int i, j, k;
    for (j = 1; j < *p_lie_num + 1; j++) {
        for (i = *p_hang_num; i > 1; i--) {
            for (k = i; k > 1; k--) {
                if (p_co->array[k][j] == 0) {
                    if (p_co->array[k - 1][j] != 0) {
                        fall_for_one_lie(p_hang_num, p_lie_num, k - 1
                    }
                    else {
                        fall_for_one_lie(p_hang_num, p_lie_num, k - 1
                    }
                }
            }
        }
        //p_co->array[1][j] = 0;
    }
    for (i = 0; i < *p_hang_num + 2; i++) {
        for (j = 0; j < *p_lie_num + 2; j++) {
            p_co->position[i][j] = '0';
        }
    }
}
```

7. 下落动画效果的实现：

首先实现一格的下落（向下打印并覆盖上方），然后实现一列的下落，最后对数组自下而上、自左向右判断是否下落。

8. 菜单项9的实现过程：

输入行列的函数->输入目标的函数->随机生成初始化数组
->设置窗口和缓冲区大小->动画打印数组->进入9专用的整合函数->结束函数

整合函数：
进入循环
{键盘鼠标控制函数->计分->判断能否继续合并->判断是否达到目标}

键盘鼠标控制函数：
进入循环
{position归零->读取键鼠->判断状态执行相应过程}
->判断菜单项为9 ->合并+下落+增补+打印

9. 覆盖新值特定概率生成的实现过程：

采用类似分数占比的方法，取总分母为一个较大的数例如10000，在0-10000中随机产生值，在某个区间中则输出特定值，区间大小占比即为概率。等概率则直接随机输出就可以。

```
else if (m == 6) {
    p = rand() % 10000 + 1;
    if (p > 0 && p <= 8000) {
        p_co->array[i][j] = rand() % (m - 2) + 1;
        p_co->position[i][j] = '*';
    }
    else if (p > 8000 && p <= 9500) {
        p_co->array[i][j] = 5;
        p_co->position[i][j] = '*';
    }
    else if (p > 9500 && p <= 10000) {
        p_co->array[i][j] = 6;
        p_co->position[i][j] = '*';
    }
}
```

4. 调试过程碰到的问题

```
struct coordinate co;
struct coordinate* p_co = &co;

for (int i = 0; i < 10; i++) {
    for (int j = 0; j < 12; j++) {
        p_co->array[i][j] = 0;
        p_co->position[i][j] = '0';
    }
}

/*行列总数*/
int hang_num, * p_hang_num = &hang_num;
int lie_num, * p_lie_num = &lie_num;

/*合成目标*/
int goal, * p_goal = &goal;

/*得分*/
int score_now = 0, * p_score_now = &score_now;
int score_sum = 0, * p_score_sum = &score_sum;

/*输入坐标的char位置*/
char char_hang, * p_char_hang = &char_hang;
char char_lie, * p_char_lie = &char_lie;

/*输入坐标的int位置*/
int int_hang, * p_int_hang = &int_hang;
int int_lie, * p_int_lie = &int_lie;

/*输入坐标的int数字*/
int object, * p_object = &object;
```

问题一：变量的传递

开始进行程序的制作时，我在main函数中定义了所需要的各种变量，然后再调用函数时直接把这些变量套进去。出现的问题是变量只作为函数形参，在函数结束时原本在函数中对变量做的操作传不出来。

解决方法是在设置变量的同时就设出其对应指针变量，函数形参设为指针型式，通过指针传参就可以实现对变量的操作，并且不用担心变量传不出来。

问题二：鼠标控制

由于对工具函数的不熟悉，当我照搬给出的键盘鼠标函数，基本可以实现想要达到的效果，但是在嵌套中我仍想再用一个小的鼠标键盘函数，自己DIV的时候就出现了一些问题：键盘部分基本正常，可是鼠标部分总是把单击左键执行成移动选项；对此进行处理后还未单击就直接跳过了鼠标判断直接执行单击左键后的选项。

这两个问题的原因一个是所给的鼠标键盘工具函数对鼠标移动的判定优先级很高（这也可能有部分原因是和缓冲区没有及时清理掉有关），导致的结果是先读取了鼠标移动并执行此分支，就进入不了单击左键的分支了。解决方法是将这几个小分支也套入一个小循环中，使之具备反复的可读取性，这样就能使得既是先读取了鼠标移动，也仍然可以循环读取鼠标状态，当单击左键是就可以及时读取并进入单击左键的分支了。

而跳过判断单击左键直接进入选项的原因则是由于类似于缓冲区的残留，导致大循环中读取的单击左键继续传入小循环之中，因而直接判断为单击左键而执行。我的解决方法是在大循环进入小循环时执行一次关闭鼠标然后再打开鼠标，通过这种方法清除掉缓冲区。

```
cct_disable_mouse();
cct_enable_mouse();

while (loop2) {
    ret = cct_read_keyboard_and_mouse(X, Y, maction
    int x0 = (X + 4) / 8, y0 = (Y + 1) / 4;
    if (ret == CCT_MOUSE_EVENT) {
        if (maction == MOUSE_LEFT_BUTTON_CLICK &&
            loop = 0;
            loop2 = 0;
        }
        else if (maction == MOUSE_RIGHT_BUTTON_CLI
            judge = '1';
            loop = 0;
            loop2 = 0;
```

问题三： 循环控制

由于我的代码中存在许多循环嵌套循环的情况，有时希望直接跳出两层循环，但break只能跳出一层，导致难以实现相对自由的循环跳出，如右图所示，每个case中的break只会跳出switch，从而使得可以继续循环回来执行菜单函数。但是如果希望退出时，case 0的break似乎也因此困于循环中。

```
char menu_choice = me
switch (menu_choice)
    case '1':
        input_hanglie
        generate_arra
        print_current
        input_and_juc
        same_choose_p
        print_choose(
        print_array_w
        need_end();
        break;
    case '2':
        input_hanglie
        generate_arra
        print_current
        input_and_juc
        same_choose_p
        print_choose(
        print_array_w
        need_end();
        break;
    case '3':
        input_hanglie
```

解决方法是引入参数控制循环的进行，例如图中所示，在相应的情况下适当改变参数即可。

```
int L = 1;
while (L) {
```

```
case '0':
    L = 0;
    break;
```

5. 心得体会

5.1. 心得体会和经验教训

变量的指针

调用函数的形参和原本函数外的变量会“脱钩”，因此在调用函数时，使用指针变量进行传值是十分有效且恰当的方法。

变量的联系

设出的许多变量在进行相应的操作之后彼此之间会有很多联系，可以抓住这些联系简化变量的调用。

结构体

结构体可以设的更大更丰富一点，把相关的变量都加入相应结构体中可以极大的简化参数的调用，使代码更加清晰。

5.2. 分解为若干小题的提示作用

分解为若干小题的提示作用很大，例如1、2特意给出了“显示内部数组”这一提示，让我们想到可以使用不止一个数组来进行记录。并且命令行的许多内部实现而不进行打印的函数都可以在接下来的伪图形界面中进行使用。难度(或者说进度)不断加深，有一种爬山的感觉，前路既是垫脚石。

5.3. 相比于汉诺塔，函数分解更合理

在函数的分类方面，本次更有规律，分为了共用函数、命令行函数和伪图形函数等，并且在头文件声明中也简单分类，更明确了一点。

但相比于汉诺塔，本次在函数上更为进步之处还是对参数的处理，相较于汉诺塔时少数的指针使用，本次基本都是指针的使用；相较于汉诺塔中大量的返回值传参，本次基本都是void函数利用指针传值，更加有效且清晰。

参数分类的函数举例

键盘鼠标函数：引入形参judge其取1、2、3分别对应7、8、9三项需求不同的键盘鼠标效果。

进入循环

->{position归零->读取键鼠->判断状态执行相应过程}

->判断judge并相应执行

包括8、9两项的部分操作如按Q或着单击右键退出时，在循环读取键鼠退出之前将judge赋为1，突出循环后若judge为1，则直接结束，若judge为2或3，则执行相应流程后再退出。

5.4. 编写相对复杂程序的心得体会

恰当地设置变量，有意识地设置指针并在函数中调用。

可以更多地使用结构体和类（当然本次作业是不允许使用类的），将变量放入其中会清晰许多。

针对循环结构注意参数的设置以进行更加自由的跳出。

针对循环结构注意及时进行初始化，防止之前的操作影响到下一次的正常执行。

6. 附件：源程序（挑选了部分关键部分）

```
char menu()
{
    srand((unsigned int)(time(0)));
    cct_cls();
    cout << "----------------------------------------" << endl
        << "1.命令行找出可合成项并标识（非递归）" << endl
        << "2.命令行找出可合成项并标识（递归）" << endl
        << "3.命令行完成一次合成（分步骤显示）" << endl
        << "4.命令行完整版（分步骤显示）" << endl
        << "5.伪图形界面显示初始数组（无分隔线）" << endl
        << "6.伪图形界面显示初始数组（有分隔线）" << endl
        << "7.伪图形界面下用箭头键 / 鼠标选择当前色块" << endl
        << "8.伪图形界面完成一次合成（分步骤）" << endl
        << "9.伪图形界面完整版(支持鼠标)" << endl
        << "----------------------------------------" << endl
        //<< "A.命令行界面（网络自动解 - 基本版）" << endl
        //<< "B.伪图形界面（网络自动解 - 基本版）" << endl
        //<< "C.命令行界面（网络自动解 - 竞赛版）" << endl
        //<< "----------------------------------------" << endl
        << "0.退出" << endl
        << "----------------------------------------" << endl
        << "[请选择:] ";
    while (1) {
        char menu_choice = _getch();
        if ((menu_choice >= 48 && menu_choice <= 57) || (menu_choi
            return menu_choice;
        }
        else if (menu_choice >= 97 && menu_choice <= 99) {
            return (menu_choice - 32);
        }
        else {
            continue;
        }
    }
}
```

```
int main()
{
    int L = 1;
    while (L) {
        struct coordinate co;
        struct coordinate* p_co = &co;

        for (int i = 0; i < 10; i++) {
            for (int j = 0; j < 12; j++) { ... }
        }

        /*行列总数*/
        int hang_num, * p_hang_num = &hang_num;
        int lie_num, * p_lie_num = &lie_num;

        /*合成目标*/
        int goal, * p_goal = &goal;

        /*得分*/
        int score_now = 0, * p_score_now = &score_now;
        int score_sum = 0, * p_score_sum = &score_sum;

        /*输入坐标的char位置*/
        char char_hang, * p_char_hang = &char_hang;
        char char_lie, * p_char_lie = &char_lie;

        /*输入坐标的int位置*/
        int int_hang, * p_int_hang = &int_hang;
        int int_lie, * p_int_lie = &int_lie;

        /*输入坐标的int数字*/
        int object, * p_object = &object;

        char menu_choice = menu();
        switch (menu_choice) {
```

```
            case '8':
                input_hanglie(p_hang_num,
                generate_array(p_hang_num,
                my_setconsoleborder(p_hang
                print_visualized_array2(p_
                choose_with_keymouse(p_han
                need_end();
                break;
            case '9':
                input_hanglie(p_hang_num,
                input_goal(p_goal);
                generate_array(p_hang_num,
                my_setconsoleborder(p_hang
                print_visualized_array2(p_
                endless_keymouse(p_hang_nu
                need_end();
            case 'A':
                break;
            case 'B':
                break;
            case 'C':
                break;
            case '0':
                L = 0;
                break;
        }
    }
    return 0;
}
```

```
#pragma once

struct coordinate {
    int array[10][12];
    char position[10][12];
};

/* 内部数组方式实现的各函数-base */
void print_current_array(int* p_hang_num, int* p_lie_num, coordinate*
void input_position(int* p_hang_num, int* p_lie_num, char* p_char_hang
void trans_position_to_int(char* p_char_hang, char* p_char_lie, int* p
void input_and_judge(int* p_hang_num, int* p_lie_num, char* p_char_han
void same_choose_plus1(int* p_hang_num, int* p_lie_num, int* p_int_han
void same_choose_plus2(int* p_hang_num, int* p_lie_num, int hang, int
void print_choose(int* p_hang_num, int* p_lie_num, int* p_int_hang, in
void print_array_with_color(int* p_hang_num, int* p_lie_num, int* p_in
char merge_confirm(int* p_hang_num, int* p_lie_num, int* p_int_hang, i
void fall(int* p_hang_num, int* p_lie_num, int* p_int_hang, int* p_int
void print_after_fall(int* p_hang_num, int* p_lie_num, int* p_int_hang
void print_after_new(int* p_hang_num, int* p_lie_num, int* p_int_hang,
void merge_for_one(int* p_hang_num, int* p_lie_num, int* p_int_hang, i
void merge_for_all(int* p_hang_num, int* p_lie_num, int* p_int_hang, c

/* 一些内部数组/图形方式公用的函数，如判断结束等-tools */
void input_hanglie(int* p_hang_num, int* p_lie_num);
void input_goal(int* p_goal);
```

```
else if (m > 6) {//大于6时的概率设置
    p = rand() % 10000 + 1;
    if (p > 0 && p <= 8000) {
        p_co->array[i][j] = rand() % (m - 3) + 1;
        p_co->position[i][j] = '*';
    }
    else if (p > 8000 && p <= 9000) {
        p_co->array[i][j] = (m - 2);
        p_co->position[i][j] = '*';
    }
    else if (p > 9000 && p <= 10000) {
        p_co->array[i][j] = rand() % 2 + m - 1;
        p_co->position[i][j] = '*';
    }
}
```

```
/*实现合并同时记分的函数*/
void merge_near(int* p_hang_num, int* p_lie_num, int* p_int_hang, int* p_int_
{
    int i, j, n = 0;
    for (i = 0; i < *p_hang_num + 2; i++) {
        for (j = 0; j < *p_lie_num + 2; j++) {
            if (i == *p_int_hang && j == *p_int_lie) {
                p_co->array[i][j] += 1;
                *p_object += 1;
            }
            else if (p_co->position[i][j] == '*') {
                p_co->array[i][j] = 0;
                n++;
            }
        }
    }
    *p_score_now = (p_co->array[*p_int_hang][*p_int_lie] - 1) * (n + 1) * 3;
    *p_score_sum += *p_score_now;
}
```

```
/*根据position数列递归判断的函数（菜单项2所要求的 递归函数） */
void same_choose_plus2(int* p_hang_num, int* p_lie_num, int hang, int lie, int* p_object, coordi
{
    if (p_co->array[hang][lie] == *p_object) {
        p_co->position[hang][lie] = '*';
    }
    if ((p_co->array[hang - 1][lie] == *p_object) && (p_co->position[hang - 1][lie] != '*')) {
        p_co->position[hang - 1][lie] = '*';
        if (hang > 0) {
            same_choose_plus2(p_hang_num, p_lie_num, hang - 1, lie, p_object, p_co);
        }
    }
    if ((p_co->array[hang][lie - 1] == *p_object) && (p_co->position[hang][lie - 1] != '*')) {
        p_co->position[hang][lie - 1] = '*';
        if (lie > 0) {
            same_choose_plus2(p_hang_num, p_lie_num, hang, lie - 1, p_object, p_co);
        }
    }
    if ((p_co->array[hang][lie + 1] == *p_object) && (p_co->position[hang][lie + 1] != '*')) {
        p_co->position[hang][lie + 1] = '*';
        if (lie < *p_lie_num) {
            same_choose_plus2(p_hang_num, p_lie_num, hang, lie + 1, p_object, p_co);
        }
    }
    if ((p_co->array[hang + 1][lie] == *p_object) && (p_co->position[hang + 1][lie] != '*')) {
        p_co->position[hang + 1][lie] = '*';
        if (hang < *p_hang_num) {
            same_choose_plus2(p_hang_num, p_lie_num, hang + 1, lie, p_object, p_co);
        }
    }
}
```

```
/*改变一块颜色的函数*/
void changcolor_for_one(int i, int j, coordinate* p_co, int fg)
{
    int x = j * 8 - 4;
    int y = i * 4 - 1;
    int bg = (p_co->array[i][j] % 14);
    cct_gotoxy(x, y);
    cct_setcolor(bg, fg);
    cout << "┌──┐";
    cct_gotoxy(x, y + 1);
    cout << "│" << setw(2) << p_co->array[i][j] << "│";
    cct_gotoxy(x, y + 2);
    cout << "└──┘";
    cct_setcolor();
}
```

```
/*继续进行需要的提示输入函数*/
void need_judge(char key1, char key2)
{
    int X = 0, Y = 0;
    int ret, maction;
    int keycode1, keycode2;
    int loop3 = 1;

    cct_disable_mouse();
    cct_enable_mouse();
    while (loop3) {
        ret = cct_read_keyboard_and_mouse(X, Y, maction, keycode1, keycode2);
        if (ret == CCT_MOUSE_EVENT) {
            switch (maction) {
                case MOUSE_LEFT_BUTTON_CLICK:
                    loop3 = 0;
                    break;
            }
        }
        else if (ret == CCT_KEYBOARD_EVENT) {
            if (keycode1 == key1 || keycode1 == key2) {
                loop3 = 0;
            }
        }
    }
}
```

装 订 线

```
if (confirm == 'Y') {
    merge_near(p_hang_num, p_lie_num, p_int_hang, p_int_
    print_array_with_color(p_hang_num, p_lie_num, p_int_
    cout << endl;
    print_current_score(p_hang_num, p_lie_num, p_int_han
    fall(p_hang_num, p_lie_num, p_int_hang, p_int_lie, p
    print_after_fall(p_hang_num, p_lie_num, p_int_hang,
    add_new(p_hang_num, p_lie_num, p_int_hang, p_int_lie
    print_after_new(p_hang_num, p_lie_num, p_int_hang, p
    cout << endl
        << "本次合成结束，按回车键继续新一次的合成..." <
    int n = 0;
    for (int i = 1; i < *p_hang_num + 1; i++) {
        for (int j = 1; j < *p_lie_num + 1; j++) {
            n += same_choose(p_hang_num, p_lie_num, i, j
        }
    }
    if (n == 0) {
        cout << "无可合并的项，游戏结束！";
        break;
    }
    int bg, fg;
    while (1) {
        char ch = _getch();
        if (ch == '\n' || ch == '\r') {
            break;
        }
    }
    if (*p_object >= *p_goal) {
        cct_getcolor(bg, fg);
        cct_setcolor(COLOR_HYELLOW, COLOR_HRED);
        cout << "已经合成到" << *p_object;
        cct_setcolor(bg, fg);
        cout << endl
            << "按回车键继续向更高目标进发..." << endl;
        (*p_goal)++;
        while (1) {
            char ch = _getch();
            if (ch == '\n' || ch == '\r') {
                break;
            }
        }
    }
```

```
/*菜单项9需要的循环整合的函数*/
void endless_keymouse(int* p_hang_num, int* p_lie_num, int* p_int_hang
{
    int lines = *p_hang_num * 4 + 7;
    int key = 1;
    while (key) {
        key = choose_with_keymouse(p_hang_num, p_lie_num, p_int_hang,
        cct_gotoxy(0, 0);
        print_current_score(p_hang_num, p_lie_num, p_int_hang, p_int_l
        int n = 0;
        for (int i = 1; i < *p_hang_num + 1; i++) {
            for (int j = 1; j < *p_lie_num + 1; j++) {
                n += same_choose(p_hang_num, p_lie_num, i, j, p_co);
            }
        }
        if (n == 0) {
            cct_setcolor(COLOR_BLACK, COLOR_HYELLOW);
            cct_gotoxy(0, lines - 4);
            cout << "无可合并的项，游戏结束！";
            cct_setcolor();
            cout << "按Q/单击鼠标左键退出
            need_judge('q','Q');
            cct_gotoxy(0, lines - 4);
            cout << "
            break;
        }
        if (*p_object >= *p_goal) {
            cct_setcolor(COLOR_BLACK, COLOR_HYELLOW);
            cct_gotoxy(0, lines - 4);
            cout << "已经合成到，" << *p_object;
            cct_setcolor();
            cout << "按回车键/单击左键继续向更高目标进发
            (*p_goal)++;
            need_judge(13);
        }
    }
    cct_gotoxy(0, lines - 4);
}
```

```
/*
键盘鼠标选择的函数
--1为菜单项7
--2为菜单项8
--3为菜单项9
*/
int choose_with_keymouse(int* p_hang_num, int* p_lie_num, int* p_int_hang, int
{
    int cols = *p_lie_num * 8 + 5;
    int lines = *p_hang_num * 4 + 7;

    int X = 0, Y = 0;
    int ret, maction;
    int keycodel, keycode2;
    int loop = 1;

    cct_setcursor(CURSOR_INVISIBLE);    //关闭光标
    cct_enable_mouse();

    int x_rem = 1, y_rem = 1;
    while (loop) {

        for (int a = 0; a < *p_hang_num + 2; a++) {
            for (int b = 0; b < *p_lie_num + 2; b++) {
                p_co->position[a][b] = '0';
            }
        }

        int t = 0;
        ret = cct_read_keyboard_and_mouse(X, Y, maction, keycodel, keycode2);
        int x = (X + 4) / 8, y = (Y + 1) / 4;
```

```
        int x = (X + 4) / 8, y = (Y + 1) / 4;
        if (ret == CCT_MOUSE_EVENT) {
            cct_gotoxy(0, lines - 4);
            cout << "[当前鼠标] ";
            switch (maction) {
                case MOUSE_ONLY_MOVED:
                    if (x > 0 && x < *p_lie_num + 1 &&
                        cout << (char)(y + 64) << "行"
                        if (x_rem != x || y_rem != y)
                            changcolor_for_one(y_rem,
                        }
                        changcolor_for_one(y, x, p_co,
                        x_rem = x;
                        y_rem = y;
                    }
                    else {
                        cout << "位置非法
                    }
                    break;
```

装 订 线

```
case MOUSE_LEFT_BUTTON_CLICK:            //按下左键
    if (x > 0 && x < *p_lie_num + 1 && y > 0 && y < *p_hang_num + 1) {
        if (judge == '1') {
            cout << "选中了" << (char)(y + 64) << "行" << x - 1 << "列
            cct_gotoxy(0, lines - 3);
            judge = '1';
            loop = 0;
        }
        else {
            *p_int_hang = y;
            *p_int_lie = x;
            *p_object = p_co->array[y][x];
            int loop2 = 1;
            if (same_choose(p_hang_num, p_lie_num, p_int_hang, p_int_lie, p_object, p_
                same_choose_plus2(p_hang_num, p_lie_num, *p_int_hang, *p_int_lie, p_ob
                merge_confirm_visualized(p_hang_num, p_lie_num, p_int_hang, p_int_lie,
                t = 1;
            }
            else {
                cct_setcolor(0, 6);
                cout << "周围无相同值，";
                cct_setcolor();
                cout << "箭头键/鼠标移动，回车键/单击左键选择，Q或单击右键结束
                break;
            }
            cct_disable_mouse();
            cct_enable_mouse();

            while (loop2) {
                ret = cct_read_keyboard_and_mouse(X, Y, maction, keycode1, keycode2);
```

```
while (loop2) {
    ret = cct_read_keyboard_and_mouse(X, Y, maction, keycode1, keycode
    int x0 = (X + 4) / 8, y0 = (Y + 1) / 4;
    if (ret == CCT_MOUSE_EVENT) {
        if (maction == MOUSE_LEFT_BUTTON_CLICK && t == 1 && x == x0 &&
            loop = 0;
            loop2 = 0;
        }
        else if (maction == MOUSE_RIGHT_BUTTON_CLICK) {
            judge = '1';
            loop = 0;
            loop2 = 0;
        }
        else if (maction == MOUSE_ONLY_MOVED && (x != x0 || y != y0))
            print_all_color(p_hang_num, p_lie_num, p_co);
            loop2 = 0;
        }
    }
    else if (ret == CCT_KEYBOARD_EVENT) {
        if (keycode1 == 224) {
            print_all_color(p_hang_num, p_lie_num, p_co);
            loop2 = 0;
        }
        else if (keycode1 == 13 && t == 1) {
            loop = 0;
            loop2 = 0;
        }
        else if (keycode1 == 'q' || keycode1 == 'Q') {
            judge = '1';
            loop = 0;
            loop2 = 0;
        }
    }
}//end_of_loop2
```

```
                }//end of swicth(keycode1)
            }//end of if(x > 0 && x < *p_lie_num + 1 && y > 0 &&
        }//end of else if(ret == CCT_KEYBOARD_EVENT)
    } //end of while(1)
    if (judge == '1') {
        cct_disable_mouse();    //禁用鼠标
        cct_setcursor(CURSOR_VISIBLE_NORMAL);   //打开光标
        cct_gotoxy(0, lines - 4);
        return 0;
    }
    else if(judge=='2') {
        merge_near(p_hang_num, p_lie_num, p_int_hang, p_int_lie,
        merge_visualized(p_hang_num, p_lie_num, p_int_hang, p_int
        cct_gotoxy(0, lines - 4);
        cout << "合成完成，回车键/单击左键下落0
        need_judge(13);
        fall_visualized(p_hang_num, p_lie_num, p_int_hang, p_int_
        cct_gotoxy(0, lines - 4);
        cout << "下落0完成，回车键/单击左键填充新值
        need_judge(13);
        add_new(p_hang_num, p_lie_num, p_int_hang, p_int_lie, p_o
        print_all_color(p_hang_num, p_lie_num, p_co);
        cct_gotoxy(0, lines - 4);
        cct_setcolor(0, 6);
        cout << "本次合成结束，按C/单击左键继续新一次的合成
        cct_setcolor();
        need_judge('c', 'C');
    }
    else if (judge == '3') {
        merge_near(p_hang_num, p_lie_num, p_int_hang, p_int_lie,
        merge_visualized(p_hang_num, p_lie_num, p_int_hang, p_int
        fall_visualized(p_hang_num, p_lie_num, p_int_hang, p_int_
        Sleep(50);
        add_new(p_hang_num, p_lie_num, p_int_hang, p_int_lie, p_o
        print_all_color(p_hang_num, p_lie_num, p_co);
    }
    cct_gotoxy(0, lines - 4);
    return 1;
}
```

```
else if (ret == CCT_KEYBOARD_EVENT) {
    cct_gotoxy(0, lines - 4);
    x = x_rem;
    y = y_rem;
    cout << "[当前键盘] ";
    if (x > 0 && x < *p_lie_num + 1 && y > 0 && y < *p_hang_num + 1) {
        switch (keycode1) {
            case 224:
                switch (keycode2) {
                    case KB_ARROW_UP:
                        //cout << "上箭头    ";
                        if (y == 1) {
                            y = *p_hang_num;
                        }
                        else {
                            --y;
                        }
                        cout << (char)(y + 64) << "行" << x - 1 << "列
                        if (x_rem != x || y_rem != y) {
                            changcolor_for_one(y_rem, x_rem, p_co, 0);
                        }
                        changcolor_for_one(y, x, p_co, 7);
                        x_rem = x;
                        y_rem = y;
                        break;
```

装 订 线

装 订 线

```
/*本小题结束的判断函数*/
void need_end()
{
    int x, y;
    char the_end[4];
    cct_getxy(x, y);
    y += 1;
    cout << endl
        << "本小题结束，请输入End继续..."
    cct_gotoxy(x + 29, y);
    while (1) {
        for (int i = 0; i < 3; i++) {
            cin >> the_end[i];
        }
        cin.clear();
        cin.ignore(65536, '\n');

        if ((the_end[0] == 'e' || the_end[0] == 'E') && (the_end[1] ==
            break;
        }
        else {
            cct_showch(x, y, ' ', COLOR_BLACK, COLOR_WHITE, 64);
            cct_showstr(0, y + 1, "输入错误，请重新输入
            cct_gotoxy(x, y);
        }
    }
}
```

```
/*打印新值填充后的数组（不同色标识）*/
void print_after_new(int* p_hang_num, int* p_lie_num, int* p_int_hang, int* p_i
{
    int i, j;
    int bg_color, fg_color;
    cout << endl
        << "新值填充后的数组(不同色标识)：" << endl
        << "  |";
    for (i = 0; i < *p_lie_num; i++) {
        cout << setw(3) << i;
    }
    cout << endl << "--+";
    for (i = 0; i < *p_lie_num; i++) {
        cout << "---";
    }
    cout << "-" << endl;
    for (i = 1; i < *p_hang_num + 1; i++) {
        cout << (char)(i + 64) << " |";
        for (j = 1; j < *p_lie_num + 1; j++) {
            if (p_co->position[i][j] == '*' && p_co->array[i][j] < 10) {
                cct_getcolor(bg_color, fg_color);
                cout << "  ";
                cct_setcolor(COLOR_HYELLOW, COLOR_BLACK);
                cout << p_co->array[i][j];
                cct_setcolor(bg_color, fg_color);
            }
            else if (p_co->position[i][j] == '*' && p_co->array[i][j] >= 10) {
                cct_getcolor(bg_color, fg_color);
                cout << " ";
                cct_setcolor(COLOR_HYELLOW, COLOR_BLACK);
                cout << p_co->array[i][j];
                cct_setcolor(bg_color, fg_color);
            }
            else {
                cout << setw(3) << p_co->array[i][j];
            }
        }
        cout << endl;
    }
```

```
/*伪图形有分隔线打印数组（菜单项6的函数）*/
void print_visualized_array2(int* p_hang_num, int* p_lie
{
    int cols = *p_lie_num * 8 + 5;
    int lines = *p_hang_num * 4 + 7;
    char char_hang = 'A';

    int x = 6, y = 1;
    int x1 = 0, y1 = y + 3;
    int x2 = 2, y2 = 2;

    cct_setcolor();//0-黑 7-白 15-亮白
    /*打横轴*/
    cct_gotoxy(x, y);
    for (int i = 0; i < *p_lie_num; i++) {
        cout << setiosflags(ios::left) << setw(8) << i;
    }
    /*打纵轴*/
    for (int i = 0; i < *p_hang_num; i++, char_hang++, y
        cct_gotoxy(x1, y1);
        cout << char_hang;
    }
```

```
/*打黑框和白底*/
cct_setcolor(15, 0);
cct_gotoxy(x2, y2);
/*首行*/
cout << "┌";
for (int i = 0; i < *p_lie_num - 1; i++) {
    cout << "────┬";
    Sleep(3);
}
cout << "────┐";
/*中间*/
for (int i = 0; i < *p_hang_num - 1; i++) {
    cct_gotoxy(x2, ++y2);
    cout << "│";
    for (int j = 0; j < *p_lie_num; j++) {
        cout << "    │";
        Sleep(3);
    }
    cct_gotoxy(x2, ++y2);
    cout << "├";
    for (int j = 0; j < *p_lie_num; j++) {
        cout << "────┼";
        Sleep(3);
    }
}
```

```
    /*尾行*/
    cct_gotoxy(x2, ++y2);
    cout << "│";
    for (int j = 0; j < *p_lie_num; j++) {
        cout << "    │";
        Sleep(3);
    }
    cct_gotoxy(x2, ++y2);
    cout << "│";
    for (int j = 0; j < *p_lie_num; j++) {
        cout << "    │";
        Sleep(3);
    }
    cct_gotoxy(x2, ++y2);
    cout << "│";
    for (int j = 0; j < *p_lie_num; j++) {
        cout << "    │";
        Sleep(3);
    }
    cct_gotoxy(x2, ++y2);
    cout << "└";
    for (int i = 0; i < *p_lie_num - 1; i++) {
        cout << "────┴";
        Sleep(3);
    }
    cout << "────┘";
    cct_setcolor();
    print_all_color(p_hang_num, p_lie_num, p_co);
    cct_gotoxy(0, lines - 4);
}
```