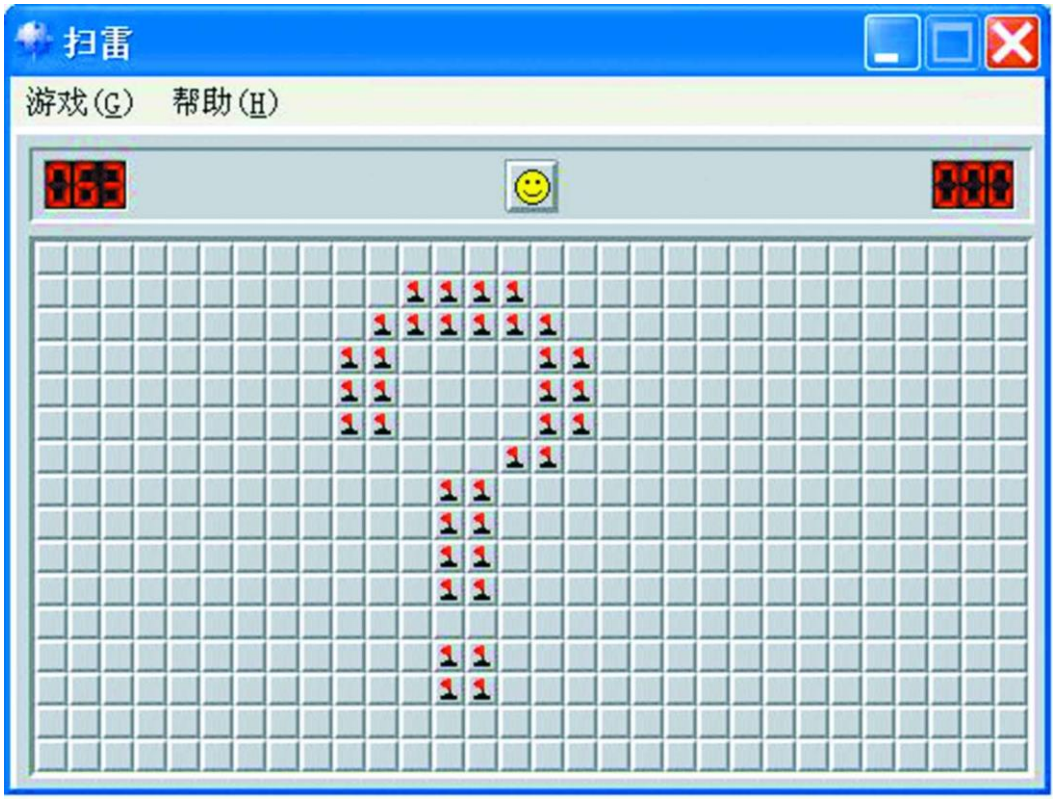
挖地雷实验报告



班级 ： 软件工程

作者姓名：陈君

学号 ： 2250420

日期 ： 1.3

扫雷游戏实验报告

2250420 陈君

一、题目及基本要求描述

1.题目描述：

设计并实现一个简单的扫雷游戏，具体要求如下：游戏板面应为一个矩形网格，包含地雷和数字，玩家需要根据数字提示推断雷区情况。实现点击方格揭示内容的功能，当点击到地雷时游戏结束。在合适的位置添加标记功能，玩家可以标记可能包含地雷的方格。游戏应该有计时器，记录玩家完成游戏所用的时间。游戏界面要友好、直观，能够清晰展示游戏的进行情况。

2.基本要求：

问题8的要求:

1、能够选择难度并显示内部数组

2、输入初始位置并显示被打开的初始区域选择难度[1..3]，打印对应大小的内部数组，要求第一次点击的位置一定不能是地雷，且这个位置周围的一圈格子内没有地雷这样可以保证第一次点击能够直接打开一片初始区域。

3、能够继续点击未知区域，如果点击到地雷就提示游戏结束

如果点击到地雷数为0的区域就将领域打开

如果点击到周围有地雷的区域就只展示这个位置

能够判断游戏是否结束：即将除了地雷的位置的所有区域都打开

添加三个特殊的功能：1、显示本局游戏已运行时间2、标记某位置为雷3、取消标记

4、拥有伪图形化的框架并且显示内部的数据

5、在为图形化的框架上移动鼠标可以判断鼠标的位置。

6、能够在伪图形化的框架上单机初始位置并显示被打开的初始区域

7、增加按ESC退出游戏的功能

问题9的要求：

在8的基础上，可显示剩余雷数、计算本局游戏已运行时间。

二、整体设计思路

功能8的设计思路：1、通过循环监听用户的鼠标操作，根据操作进行相应的处理。2、判断游戏是否结束，若结束则显示相应的信息。3、不断更新并展示雷区的状态，提供良好的用户体验。具体实现：

输出提示信息，告知用户按Esc键可以退出游戏。

调用putoutframe函数输出雷区的框架。

调用setblocks函数为方块涂色，显示雷区的当前状态。

进入一个循环，等待用户的操作。

在循环中，获取用户鼠标的操作信息（左键单击、右键单击、Esc键）。

如果是左键单击，根据鼠标点击的位置执行相应操作：

如果是第一次点击，调用create_bomb2函数创建雷区数组。

如果点击到地雷，显示游戏失败信息，涂色显示所有地雷位置。

如果点击到数字0，进行领域展开操作。

如果点击到数字1-8，标记该位置。

如果是右键单击，进行标记和取消标记的操作，同时更新雷的数量。

如果是Esc键，退出游戏。

检查游戏是否胜利，如果是，显示胜利信息并退出游戏。

每次循环结束后，调用setblocks和setbombflag函数刷新雷区显示。

功能9的设计思路：

在功能8的基础上增加计时功能：计时功能：

使用LARGE_INTEGER类型的变量tick、begin、end记录计时信息。

通过QueryPerformanceFrequency和QueryPerformanceCounter函数获取计时器的频率和初始计数值。

计时信息显示：

在每次空格键按下时，通过QueryPerformanceCounter获取当前硬件计数器计数值。

计算并输出当前游戏运行时间（从游戏开始到当前时间的差值）。

三、主要功能的实现

1、void setparameter(int* B_row, int* B_column, int* bombnum, int difficulty)//设置参数函数

具体实现：通过判断difficulty的不同，设置不同的游戏参数：

如果difficulty为1，将*bombnum设为10，*B_row设为9，*B_column设为9。

如果difficulty为2，将*bombnum设为40，*B_row设为16，*B_column设为16。

如果difficulty为3，将*bombnum设为99，*B_row设为16，*B_column设为30。

2、void openarea(plaid bomb[][30], int row, int column, int B_row, int B_column);//领域展开doge

具体实现：递归调用函数:调用find(bomb, row, column, B_row, B_column);//找出0的路径和openflag(bomb, B_row, B_column)；这两个函数找出和当前点击位置的0连着的0以及0周围的flag。

3、void find(plaid bomb[][30], int row, int column, int B_row, int B_column)

这个函数的目的是进行递归地领域展开，寻找并标记与指定位置相邻的所有空白方块。

具体实现：

首先进行边界判断，如果当前处理的方块越界，就直接返回。

如果当前方块的元素不是'0'，或者已经被标记(flag为1)，则返回。

如果当前方块是'0'且未被标记，将其标记为已访问（flag设为1）。

对当前方块的上、下、左、右四个相邻方块递归调用find函数，实现对相邻空白方块的领域展开。

4、void openflag(plaid bomb[][30], int B_row, int B_column)

具体实现：

使用两层嵌套循环遍历整个雷区，检查每个方块的内容。

对于每个方块，调用bomb_checkzero函数检查是否为零。

如果是零，则将当前方块的flag属性标记为1，说明这个位置的方块后面需要上色

5、void setblocks(plaid bomb[][30], int B_row, int B_column);//伪图形界面模式下给棋盘上色

具体实现：
对于每个方块，根据其flag属性选择合适的颜色，并调用blockgetcolor函数设置颜色。
同时，将already flag属性设置为1，表示该方块已经被处理。
颜色设置：
调用blockgetcolor函数设置方块的颜色。
blockgetcolor函数的参数是方块的行索引、列索引以及颜色代码。
already flag属性：
already flag属性用于标记该方块是否已经被处理过，避免重复处理，出现棋盘刷新的bug。

6、bool is_over(plaid bomb[][30], int B_row, int B_column);\\判断胜利函数

具体实现：
使用两层嵌套循环遍历整个雷区。
对于每个方块，检查其element和flag属性。
如果存在一个非地雷方块且未被标记，返回false。
如果所有非地雷方块都被标记，返回true，表示游戏结束。

7、int getpisition(plaid bomb[][30], int bombnum, int B_row, int B_column, int& nrow, int& ncolumn,int func)//监听用户的键鼠操作并返回

具体实现：
进入一个循环，通过cct_read_keyboard_and_mouse函数获取键鼠事件。
在循环中，获取鼠标坐标，并通过changetoinside函数将坐标转换为雷区内的行列位置。
在屏幕上显示当前光标位置信息。
如果检测到鼠标左键点击，返回MOUSE_LEFT_BUTTON_CLICK，表示左键点击事件。
如果检测到鼠标右键点击且func为8，返回MOUSE_RIGHT_BUTTON_CLICK，表示右键点击事件。
如果检测到键盘事件且func为8，返回键盘按键的ASCII码。
总体思路：
通过循环不断获取鼠标和键盘事件，实时显示当前光标位置信息。
根据事件类型和功能，返回相应的值。

8、void setconsoleborder(int difficulty)//根据难度设置窗口参数

具体实现：
使用switch语句根据不同的难度级别进行不同的设置。
每个case中使用cct_setfontsize设置字体的名称、大小和加粗属性。
使用cct_setconsoleborder设置控制台的边框大小。
根据difficulty的不同，设置不同的字体和控制台边框大小。
每个case中包含了对应难度级别的具体设置。

9、void putoutframe(int B_row, int B_column)//输出图形框架

具体实现：
输出列号：使用循环打印每一列的索引。
调用putoutframelines函数打印边框线。
使用两层嵌套循环，外层表示每一行，内层表示每一行的不同层。
在每一行内，使用cout输出行号和框架线。
根据当前行的不同，调用putoutframelines函数打印不同的框架线。
putoutframelines函数：该函数用于输出不同类型的框架线，根据传入的参数类型选择打印不同的线。
cct_setcolor函数：该函数用于设置控制台输出的颜色。

10、void putoutframelines(int i, int B_column)// 根据指令i逐行输出框架的指定行

具体实现：

根据传入的参数i的值，选择不同的case，执行相应的框架线输出。

对于表头、第二层、表尾，使用for循环打印相应数量的字符，构成相应的框架线。

通过调用cct_setcolor函数设置不同的颜色。

cct_setcolor函数：

该函数用于设置控制台输出的颜色。

11、bool bomb_checkzero(plaid bomb[][30], int i, int j, int B_row, int B_column)// 检查指定位置周围是否存在被标记为打开且元素为 '0' 的方块

外层循环遍历行，内层循环遍历列，以当前位置为中心，检查周围的8个方向。

对于每一个周围的方块，判断是否满足条件（'0' 且已经被打开）。

如果存在满足条件的方块，则返回 true，表示周围有打开的 '0'。

如果遍历完整个周围都没有满足条件的方块，则返回 false。

返回一个布尔值，表示指定位置周围是否存在被标记为打开且元素为 '0' 的方块。

四、调试过程碰到的问题

1.问题一

在完成图形解的过程中发现我写出来的程序，每次点击都会刷新已经存在的flag数字。

Bug发生原因：经过仔细研读我给所有flag上色的函数我发现是由于每次点击都会执行生成框架然后再次重新全部上色，这样就会出现反复刷新的bug。

解决办法：将地雷的内部二维数组的元素设置为一个结构体，里面加上already参数代表已经上色，下次就不用上色了。然后在每次点击后，不用再重复打印框架了，直接就每次给还没有上过颜色的位置上色，已经还没有上过色的格子进行上色，经过一系列的处理最后我解决了这个bug。

2.问题二

在实现内部数组的解的时候，开始的架构想的过于简单，直接套用之前作业中的代码，导致做到后面几个小题时发现无法通过修改之前的代码完成题目的要求。

解决方法：发生错误的原因是由于我开始的时候套用之前的代码，二维数组里面的元素仅仅是一个int类型的数字无法实现字符的刷新，或者其他的字符，所以我重新书写了我的代码逻辑，将数组元素换成了结构体struct plaid {

```
char element;//扫雷图中的元素
int flag;//这个元素是否显示
```

```
    int alreadyflag;
};
```

这样包含的信息更多，为后面代码的实现带来了很大的方便。

3.问题三

在做图形解的时候，读取用户的键鼠输入出现错误且上色的位置计算繁复。

解决方法：认真研读了老师给出的控制台文件的函数介绍，并且看了demo的源码后，掌握了如何使用cct函数获取用户的键鼠输入。至于位置的计算，我书写了两个工具函数，输入内部数组的位置，能够返回不同难度下实际应该上色的位置，这个函数使用频率很高，极大的提高了代码复用率。

五、心得体会

5.1 完成本次作业得到的一些心得体会，经验教训:

1、良好的架构设计： 在编写代码之前，对整体架构进行良好的设计是非常重要的。在实现复杂功能时，需要更灵活和细致的架构，以便在后续的扩展和修改中能够更容易地适应需求变化。

2、不宜贪图便利： 有时候为了追求便捷可能选择直接套用之前的代码，但要慎重考虑是否符合新的需求，以免后期出现问题。

3、深入理解问题： 在编写代码之前，充分理解题目的要求和期望，避免过于简单地套用之前的代码而导致后期问题的出现。。

4、及时调整计划： 如果在编写代码的过程中发现之前的计划无法满足新的要求，及时调整计划，避免浪费过多的时间在错误的方向上。比如我这次由于套用之前的代码，写到后面写崩了，为了避免最后写成屎山代码，我果断调整计划，重新开始写代码的逻辑。

5.2 我在完成过程中确实考虑了前后小题的关联，而且也有效利用了前面小题的代码

比如在做图形解的时候，内部数组的位置和屏幕上实际的位置的转换计算十分繁复而且使用的频率很高所以我书写了两个工具函数，输入内部数组的位置，能够返回不同难度下实际应该上色的位置，这个函数使用频率很高，极大的提高了代码复用率。

比如在做问题8和问题9时我会有意识的使用前面几个子问题的函数，比如打印框架，比如获取用户的输入等等的函数均在多个主函数中有反复的调用，大大提高了代码的复用率，减少了代码的长度提高了代码的可读性。

同时有了上一次汉诺塔的编程训练之后我在书写前面几个简单小题的过程中我也会有意识的将代码拆分成若干个更加简单的函数这样方便后面如果有类似的功能我能够直接调用，这样子的写法也大大提高了我的代码的复用率，减少了代码的长度，提高了代码的可读性。

5.3如何才能更好的重用代码？

模块化设计： 将代码划分成独立的模块或函数，每个模块负责特定的功能。这样的设计使得每个模块都可以被单独使用，提高了代码的可重用性。

良好的函数和方法设计： 设计函数和方法时，尽量保持单一职责原则，一个函数只完成一个特定的任务。这样的函数更容易在不同的上下文中被重用。

参数化和配置： 使函数和模块可以通过参数进行配置，而不是硬编码特定的数值或设置。这样的设计使得代码更加灵活，适应不同的需求。

标准化接口： 如果有多个模块需要协同工作，定义清晰的接口规范。这样可以确保各个模块之间的兼容性，方便替换和重用。

尽量避免全局变量： 全局变量会增加代码的耦合度，降低了模块的独立性。尽量使用函数参数和返回值传递信息，减少对全局状态的依赖。

文档化： 对于被设计为通用工具或库的代码，提供清晰的文档是至关重要的。文档应该包括使用方法、参数说明和示例用法，以便其他开发人员能够理解和正确使用代码。

单元测试： 编写有效的单元测试可以确保代码的正确性和可靠性。当修改或重用代码时，有一套完善的单元测试可以帮助快速发现问题，同时也增加了代码的可维护性。

附件：源程序

主函数：

```
int main()
{

    while (true)
    {
        /* demo中首先执行此句，设置窗口大小和缓冲区*/
        setconsoleborder(4);
        int command, difficulty;
        plaid bomb[16][30] ;
        //数组初始化
        for (int i = 0; i < 16; i++)
            for (int j = 0; j < 30; j++)
            {
                bomb[i][j].element = 0;
                bomb[i][j].flag = 0;
                bomb[i][j].alreadyflag = 0;
            }
        int bombnum, B_row,B_column;
        command = showmenu();//调用menu中的展示UI
        if (command == 0)
            exit(1);
        difficulty = getdifficulty();//获取难度值
        setparameter(&B_row, &B_column, &bombnum, difficulty);
        switch (command)
        {
             case 1:
```

```
                Func1(bomb,bombnum,B_row,B_column);
                break;
            case 2:
                Func2(bomb, bombnum, B_row, B_column);
                break;
            case 3:
                Func3(bomb, bombnum, B_row, B_column);
                break;
            case 4:
                Func4(bomb, bombnum, B_row, B_column);
                break;
            case 5:
                setconsoleborder(difficulty);
                Func5(bomb, bombnum, B_row, B_column);
                break;
            case 6:
                setconsoleborder(difficulty);
                Func6(bomb, bombnum, B_row, B_column);
                break;
            case 7:
                setconsoleborder(difficulty);
                Func7(bomb, bombnum, B_row, B_column);
                break;
            case 8:
                setconsoleborder(difficulty);
                Func8(bomb, bombnum, B_row, B_column);
                break;
            case 9:
                setconsoleborder(difficulty);
                Func9(bomb, bombnum, B_row, B_column);
                break;
        }
        cout << endl << endl << "按回车键继续...";
        while (_getch() != '\r')
            ;
        cct_setcolor();
        cct_cls();
    }
    return 0;
    }
```

重要实现函数：

```
/****************************************************************************
  函数名称：
  功    能：问题8
  输入参数：
  返 回 值：
  说    明：
****************************************************************************/
void Func8(plaid bomb[][30], int bombnum, int B_row, int B_column,int command)
{
    /* 此处是你的程序开始 */
    cout << "按esc退出 ";
    putoutframe(B_row, B_column);//输出框架
```

```
    setblocks(bomb, B_row, B_column);//为方块涂色
    int time = 0;
    while (1)
    {
        int row, column;
        int ret = getpisition(bomb, bombnum, B_row, B_column, row, column, 8) ;
        if (ret == MOUSE_LEFT_BUTTON_CLICK)//如果是单击
        {
            if (time == 0)//如果是第一次点击，就创建数组
                create_bomb2(bomb, bombnum, B_row, B_column, row, column);//创建数组
            time++;
            //setbombflag(bomb, B_row, B_column, 1);
            if (bomb[row][column].element == '*'&& bomb[row][column].flag!=-1)
            {
                bomb[row][column].flag = 1;
                int nrow = 3 * (B_row)+4, ncolumn = 0;
                cct_gotoxy(ncolumn, nrow);
                cout << "你输了游戏结束       " << endl;
                setblocks(bomb, B_row, B_column);//为方块涂色
                setbombflag(bomb, B_row, B_column, 0);
                cct_gotoxy(ncolumn, nrow);
                return;
            }
            else if (bomb[row][column].element == '0')//如果为0，那么请领域展开
                openarea(bomb, row, column, B_row, B_column);//领域展开doge
            else if (bomb[row][column].element <= '8' && bomb[row][column].element >= '1')//如果只是
一个就给他涂色5
                bomb[row][column].flag = 1;
        }
        else if (ret == MOUSE_RIGHT_BUTTON_CLICK)//如果是右键
        {
            if (bomb[row][column].flag == -1)
            {
                bomb[row][column].flag = 0;
                bombnum++;
            }
            else if (bomb[row][column].flag == 0)
            {
                bomb[row][column].flag = -1;
                bombnum--;
            }
        }
        else if (ret == 27)//如果是esc,退出
        {
            return;
        }
        if (is_over(bomb, B_row, B_column))
        {
            int row = 3 * (B_row)+4, column = 0;
            cct_gotoxy(column, row);
            cout << "恭喜胜利，游戏结束" << endl;
            return;
        }
        setblocks(bomb, B_row, B_column);//为方块涂色
```

```
        setbombflag(bomb, B_row, B_column, 0);
    }
    }
/*****************************************************************************
  函数名称:
  功    能:问题九
  输入参数:
  返 回 值:
  说    明:
*****************************************************************************/

void Func9(plaid bomb[][30], int bombnum, int B_row, int B_column)
{
    LARGE_INTEGER tick, begin, end;

    QueryPerformanceFrequency(&tick);   //获得计数器频率
    QueryPerformanceCounter(&begin);    //获得初始硬件计数器计数
    /* 此处是你的程序开始 */
    int time = 0;
    cout << "Esc退出,空格显示时间";
    putoutframe(B_row, B_column);//输出框架
    setblocks(bomb, B_row, B_column);//为方块涂色
    while (1)
    {
        int row, column;
        int ret = getpisition(bomb, bombnum, B_row, B_column, row, column, 8);
        if (ret == MOUSE_LEFT_BUTTON_CLICK)//如果是单击
        {
            if (time == 0)//如果是第一次点击，就创建数组
                create_bomb2(bomb, bombnum, B_row, B_column, row, column);//创建数组
            time++;
            if (bomb[row][column].element == '*' && bomb[row][column].flag != -1)
            {
                bomb[row][column].flag = 1;
                int nrow = 3 * (B_row)+4, ncolumn = 0;
                cct_gotoxy(ncolumn, nrow);
                cout << "你输了游戏结束       " << endl;
                setblocks(bomb, B_row, B_column);//为方块涂色
                setbombflag(bomb, B_row, B_column, 0);
                cct_gotoxy(ncolumn, nrow);
                return;
            }
            else if (bomb[row][column].element == '0')//如果为0，那么请领域展开
                openarea(bomb, row, column, B_row, B_column);//领域展开doge
            else if (bomb[row][column].element <= '8' && bomb[row][column].element >= '1')//如果只是
一个就给他涂色5
                bomb[row][column].flag = 1;
        }
        else if (ret == MOUSE_RIGHT_BUTTON_CLICK)//如果是右键
        {
            if (bomb[row][column].flag == -1)
            {
                bomb[row][column].flag = 0;
```

```
                bombnum++;
            }
            else if (bomb[row][column].flag == 0)
            {
                bomb[row][column].flag = -1;
                bombnum--;
            }
            putbombnum(bombnum);
        }
        else if (ret == 27)//如果是esc,退出
        {
            return;
        }
        else if (ret == 32)//如果是空格
        {
            QueryPerformanceCounter(&end);      //获得终止硬件计数器计数
            cct_gotoxy(0, 0);
            cct_setcolor(0, 6);
            cout << "当前时间：" << setiosflags(ios::fixed) << setprecision(6) <<
double(end.QuadPart - begin.QuadPart) / tick.QuadPart << "秒";
            cct_setcolor();
            cout<<", Esc退出, 空格显示时间    " << endl;
        }
        if (is_over(bomb, B_row, B_column))
        {
            QueryPerformanceCounter(&end);      //获得终止硬件计数器计数
            cct_setcolor(0, 6);
            cout << "共用时：" << setiosflags(ios::fixed) << setprecision(6) << double(end.QuadPart
- begin.QuadPart) / tick.QuadPart << "秒";
            cct_setcolor();
            cout << "恭喜胜利，游戏结束    ";
            return;
        }
        setblocks(bomb, B_row, B_column);//为方块涂色
        setbombflag(bomb, B_row, B_column, 0);
    }
}
/*****************************************************************************
  函数名称：
  功    能：寻找0的路径
  输入参数：
  返 回 值：
  说    明：
*****************************************************************************/
void find(plaid bomb[][30], int row, int column, int B_row, int B_column)
{
    if (row<0 || row>B_row || column<0 || column>B_column)//如果数组越界就结束
        return;
    if (bomb[row][column].element != '0' || bomb[row][column].flag == 1)
        return;
    else
    {
            bomb[row][column].flag = 1;
    }
```

```
    find(bomb, row + 1, column, B_row, B_column);
    find(bomb, row, column + 1, B_row, B_column);
    find(bomb, row, column - 1, B_row, B_column);
    find(bomb, row - 1, column, B_row, B_column);
   }
/*****************************************************************************
  函数名称:
  功    能:显示光标位置,并且在点击后返回事件，并且更新鼠标所指的数组坐标
  输入参数:
  返 回 值:
  说    明:
*****************************************************************************/
int getpisition(plaid bomb[][30], int bombnum, int B_row, int B_column, int& nrow, int& ncolumn,int
func)
{
    int row = 3 * (B_row)+4, column = 0;
    int loop = 1;
    cct_enable_mouse();
    cct_setcursor(CURSOR_INVISIBLE);
    while (loop)
    {
        int x = 0;
        int y = 0;//鼠标坐标
        int maction, kaction1, kaction2, ret;//键鼠事件
         ret = cct_read_keyboard_and_mouse(x, y, maction, kaction1, kaction2);
        if (ret == CCT_MOUSE_EVENT) {
            cct_gotoxy(column, row);

            changetoinside(nrow, ncolumn, B_row, B_column, x, y);
            if (nrow >= 0 && ncolumn >= 0)
                cout << "[当前光标] " << char(nrow + 'A') << "行" << ncolumn << "列       ";
            else
                cout << "[当前光标] 位置非法     ";
            if (maction == MOUSE_LEFT_BUTTON_CLICK)//按下左键,退出循环
                return MOUSE_LEFT_BUTTON_CLICK;
            if (maction == MOUSE_RIGHT_BUTTON_CLICK && func == 8)
                return MOUSE_RIGHT_BUTTON_CLICK;
        }
        else if (ret == CCT_KEYBOARD_EVENT&&func==8)
        {
            return  kaction1;
        }
    }
}
```