Tongji University

结构体与引用的简单应用

——以扫雷演示程序为例

朱泽凯 2051991 智交

2020-12-12

1. 题目

利用所给的 cct_ 工具库实现 cmd 下的扫雷游戏演示，要求用数子显示和伪图形界面显示两种方式，以分小题的方式逐步实现新建雷图、打开雷区、标记雷区（插旗）、检查区域以及游戏状态判定、游戏时间记录等功能。

具体小题要求如下：

（数组模式）

1. 选择难度并显示内部数组(提供三种难度，地图大小不同，雷数不同，随机生成一定数量的雷，并把每个区域的雷相关信息输出，是雷画’*’，不是雷输出周围八格雷的个数)

2. 输入初始位置并显示被打开的初始区域（难度选择、信息输出与前相同，但雷区要求第一次选择的位置不是雷）

3. 内部数组基础版（实现扫雷游戏的基本运行，其它要求与前相同，增加后续的点开功能以及正确判断游戏状态）

4. 内部数组完整版（其它与前相同，增加后续的检查、标记功能）

（伪图形界面模式）

5. 画出伪图形化的框架并显示内部数据(除了输出形式变为伪图形界面,内部数组0不显示，其它与2相同）

6. 在伪图形化的框架上移动鼠标，判断鼠标的位置（其它要求与6相同）

7. 用鼠标在伪图形化的框架上单击初始位置并显示被打开的初始区域（除了操作信息的获取方式变为鼠标以及输出形式为伪图形界面，其它与2相同）

8. 伪图形化游戏基础版（除了操作信息的获取方式变为鼠标以及输出形式为伪图形界面，其它与3相同）

9. 伪图形化游戏完整版（除了操作信息的获取方式变为鼠标以及输出形式为伪图形界面，其它与4相同）

2. 整体设计思路

课程解锁了结构体与引用两个工具，于是这次实践尝试利用它们达到简化与稳定代码的功能。

注意到之前的大小作业代码中，需要的时候，函数总是要写许多的参变量，这样在写的过程中出错概率一定程度上提升了。若将所有与演示相关的必要信息数据用一个结构体封装起来，函数传参时就只需要写一个参数了，这样代码易读易写，出错的概率也少了，对于代码维护上也是很有帮助。

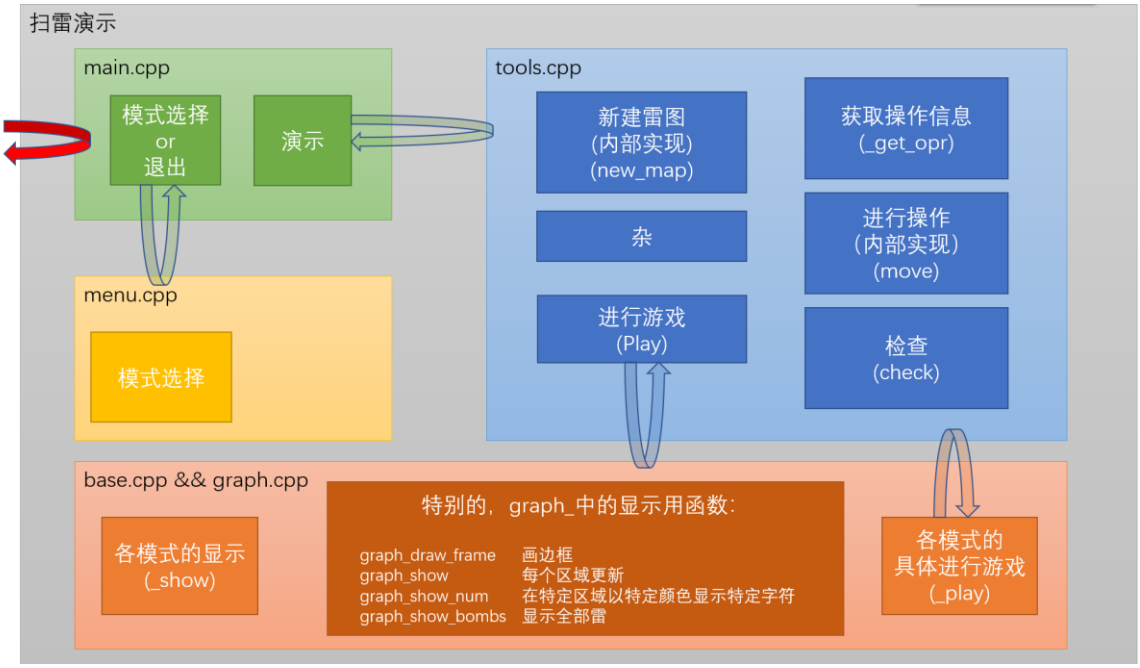
用一个总的结构体来存信息，空间上变大了，整体复制等操作所需时间也变长了。这时又可以运用引用，在函数传参过程中直接使用已有变量，避免过多冗余的空间时间消耗。

具体地，这个结构体中包含内容：

- 一个数组记录地图的雷信息（以1为起始有效编号，值为-1代表是雷，值为0~8代表周围的雷数）
- 一个数组记录地图的标记信息（以1为起始有效编号，值为0代表未点开，值为1代表插了旗即标记了，值为2代表点开了即需要显示出来）
- 一些单独的整型变量，记录模式、难度、地图长、地图宽、总雷数、当前标记数等信息

还有合理利用 const 关键字也可以让函数目的性更强、降低出错率。

3. 主要功能的实现



4. 调试过程碰到的问题

滚轮滚动的时候返回给坐标变量的值并非鼠标所在的位置信息对应的值。
解决方法：忽略滚轮操作信息。

5. 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

真正写代码之前的预先计划设计和写代码一样重要，合理的准备工作（如程序框架、函数分类、易错点预知等）往往能显著地提高写代码以及调试的效率。

5.2. 分为若干小题的方式是否对你的设计起到了一定的提示作用

是。这几个小题对整个程序的框架构建有一定的提示作用。

5.3. 与汉诺塔相比，你在函数的分解上是否做到了更合理？介绍一下你的几个重点函数的分类方法及使用情况

“更合理”不敢说。

分类方法大部分都是直接引用传参一个总的 GMDATA 结构体，函数内部再根据结构体中的信息进行相应功能的分类实现，例如附件中的 play_graph 函数。还有一个函数 newmap ，就没有用到 GMDATA 结构体，我认为它应该只负责按需新建一个符合要求的雷区信息数组，于是就只传了一个数组和起始点的坐标。

更具体的函数分类方法见“3.主要功能实现”中的图。

其实还有个想法，本次实践中由于代码量较少，而且功能也较少，而且相对琐碎（又有时间记录、还有三四种操作可供选择），所以我是根据 GMDATA 结构体中的 mode 一个参数，在函数里进行判断的，所以有较多的 if 判断语句。其实还有一种方法，就是结构体中为每一种操作设一个成员参数，函数实现的时候只根据对应的那个参数实现即可。后者的方案对于以后的大代码可能更实用。

装

订

线

5.4.以本次作业为例，谈谈编写一个相对复杂的程序的心得体会

1. **模块化编程思路**

   和上次汉诺塔大作业一样，先做好大致的模块规划，再进行具体编程，工作效率会高很多。

2. **将琐碎的信息集成起来**

   如这次，我就将各函数所需的必要信息都提炼成参变量封装在了一个叫 GMDATA 的结构体中，又将每个操作的信息封装在了一个叫 OPERATION 的结构体中。（见下图1）

```
struct GMDATA
{
    int mode;
    int level;
    int Xlen, Ylen;
    int bombnum;
    int map[Maxlen][Maxlen];    //下标从 1 开始
    int tag[Maxlen][Maxlen];    //0:未开封  1:插旗  2:显示
    double start_time;
    int flagnum;
};
 //(double)(finish - start) / CLOCKS_PER_SEC

struct OPERATION
{
    int mode=0; //0:打开  1:插/收旗  2:运行时间  3:退出
    int X=0, Y=0;
};
```

图1

这样写的好处在本次实践中就颇有体现。不仅写函数时无需过多考虑需要定义多少参变量，而且该结构体的内容还可以随工程逐步推进而进行更改，同时之前的函数也无需进行更改也能灵活的使用更改了的结构体。

3. **合理操作以减少不必要的时间、空间消耗**

例如函数传参时利用引用，这样就可以不使用全局变量而实现类似全局变量的效果了。

4. **运用 const 关键字**

由于函数传参时利用了引用，使得其直接使用了实参的地址，操作时要十分小心，一旦出bug就会影响到函数外部的那个结构体变量，又可能影响到整个程序。在形参是属于“固定属性”性质的信息时，可以利用 const 关键字锁定它，防止函数中错误书写导致意想不到的后果。

装

6. 附件：源程序

（篇幅原因只附部分程序）

订

线

```
/* 伪图形界面模式 */
void play_graph(GMDATA& gmdata)
{
    OPERATION opr;

    if (gmdata.mode == 5 || gmdata.mode==6) {
        for (int i = 1; i <= gmdata.Xlen; i++)
            for (int j = 1; j <= gmdata.Ylen; j++)
                gmdata.tag[i][j] =2;
        newmap(gmdata.map, gmdata.Xlen, gmdata.Ylen, gmdata.bombnum);
        graph_draw_frame(gmdata);
        graph_show(gmdata);
    }
    if (gmdata.mode == 6) {
        for (opr.mode=-1; opr.mode!=OPR_OPEN; )
            get_graph_opr(gmdata, opr);
        return;
    }

    if (gmdata.mode >= 7 && gmdata.mode <= 9) {         //init map
        for (int i = 1; i <= gmdata.Xlen; i++)
            for (int j = 1; j <= gmdata.Ylen; j++)
                gmdata.tag[i][j] = gmdata.map[i][j] = 0;
        gmdata.flagnum = 0;

        graph_draw_frame(gmdata);

        for (get_graph_opr(gmdata, opr); opr.mode != OPR_OPEN; get_graph_opr(gmdata, opr)) {
            if (opr.mode == OPR_EXIT) {
                return;
            }
            if (opr.mode == OPR_TIME) {
                cct_gotoxy(0, 0);
                puts("还没开始呢            ");
                Sleep(500);
                cct_gotoxy(0, 0);
                puts("                      ");
                continue;
            }
            else {
                move(opr, gmdata);
                graph_show(gmdata);
            }
        }

        newmap(gmdata.map, gmdata.Xlen, gmdata.Ylen, gmdata.bombnum, opr.X, opr.Y);//新建雷图，开始计时
        gmdata.start_time = double(clock());

        move(opr, gmdata);
        graph_show(gmdata);

        if (gmdata.mode == 9) {
            cct_gotoxy(0, 0);
            puts("开始计时                  ");
        }
    }
    if (gmdata.mode == 8 || gmdata.mode == 9) {
```

装 订 线

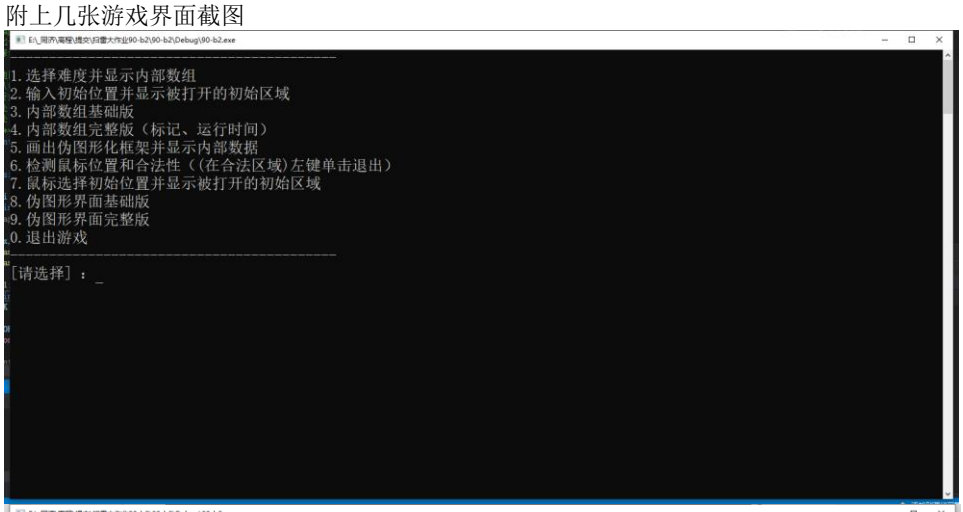
```
        int tmp;
        while ((tmp = check(gmdata)) == 0) {

            get_graph_opr(gmdata, opr);

            if (opr.mode == OPR_EXIT)
                return;

            move(opr, gmdata);
            graph_show(gmdata);

        }

        if (tmp == 1) {         //赢了

            cct_gotoxy(0, 0);

            cout << "赢了  " << "用时(秒)" << double((clock() - gmdata.start_time) / CLOCKS_PER_SEC)
<< endl;
        }
        else if (tmp == 2) {    //输了

            graph_show_bombs(gmdata);

            cct_gotoxy(0, 0);

            puts("输了                          ");
        }
    }

}

/*
opr.mode
        0:打开
        1:插/收旗
        2:运行时间
        3:退出

*/
void move(OPERATION opr, GMDATA &D)
{
    if (opr.mode == 3)
        return;


    if (opr.mode == 0 && D.tag[opr.X][opr.Y]!=2) {      //打开

        if (D.tag[opr.X][opr.Y] == 1)    //消去旗子
            D.flagnum--;

        D.tag[opr.X][opr.Y] = 2;

        if (D.map[opr.X][opr.Y] == 0) {
            for (int X = opr.X, Y = opr.Y, f = 0; f < 8; f++) {
                opr.X = X + fx[f][0];
                opr.Y = Y + fx[f][1];
```

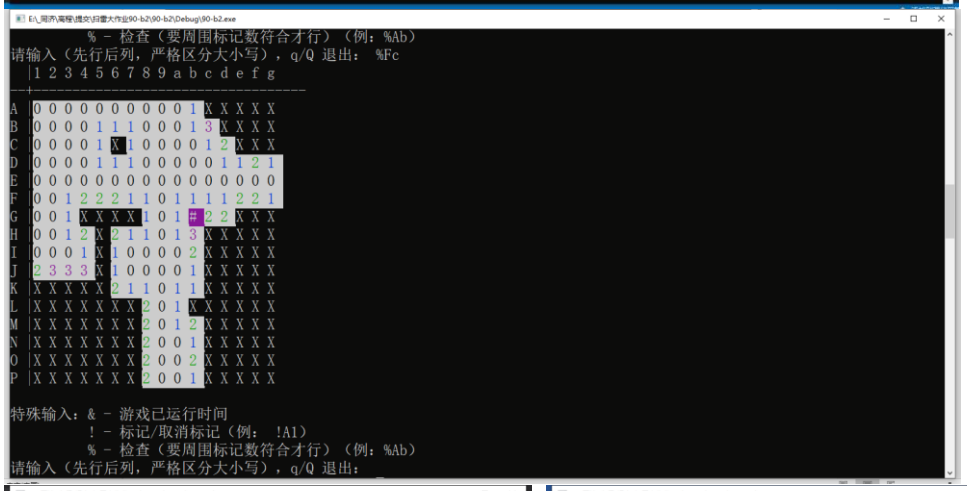
```
                if (opr.X<1 || opr.X>D.Xlen || opr.Y<1 || opr.Y>D.Ylen)
                    continue;

                if (D.tag[opr.X][opr.Y] != 2)
                    move(opr,D);       //dfs
            }
        }

    }
    if (opr.mode == 1 && D.tag[opr.X][opr.Y]!=2) {      //插/收旗
        D.tag[opr.X][opr.Y] = (D.tag[opr.X][opr.Y] + 1) % 2;

        if (D.tag[opr.X][opr.Y] == 0)
            D.flagnum--;
        else if (D.tag[opr.X][opr.Y] == 1)
            D.flagnum++;
    }
    if (opr.mode == 2 && (D.mode==4 || D.mode==9)) {         //运行时间
        double tmp = clock();
        puts("");

        if (D.mode == 8 || D.mode == 9)
            cct_gotoxy(0, 0);

        cout << "已运行时间  " << setiosflags(ios::fixed) << setprecision(3) << ((double)(tmp -
D.start_time) / CLOCKS_PER_SEC) << "秒                        " << endl;
    }

    if (opr.mode == 4 && D.tag[opr.X][opr.Y]==2) {      //打开过了才检查
        int tmp=0;
        OPERATION tmpopr;
        for (int i = 0; i < 8; i++)       //统计周围标记数
            tmp += (opr.X>=1 && opr.X<=D.Xlen && opr.Y>=1 && opr.Y<=D.Ylen && D.tag[opr.X +
fx[i][0]][opr.Y + fx[i][1]] == 1);

        if (tmp == D.map[opr.X][opr.Y]) {     //周围标记数符合才检查
            for (int i = 0; i < 8; i++) {
                tmpopr.X = opr.X + fx[i][0];
                tmpopr.Y = opr.Y + fx[i][1];

                if (tmpopr.X<1 || tmpopr.X>D.Xlen || tmpopr.Y<1 || tmpopr.Y>D.Ylen)        //确
保在界内
                    continue;

                if (D.tag[tmpopr.X][tmpopr.Y] == 0) {
                    tmpopr.mode = OPR_OPEN;          //检查就意味着周围不是雷的都可以打开
                    move(tmpopr, D);
                    //tmpopr.mode = OPR_CHEK;
                    //move(tmpopr, D);

                }
            }

        }
    }
```

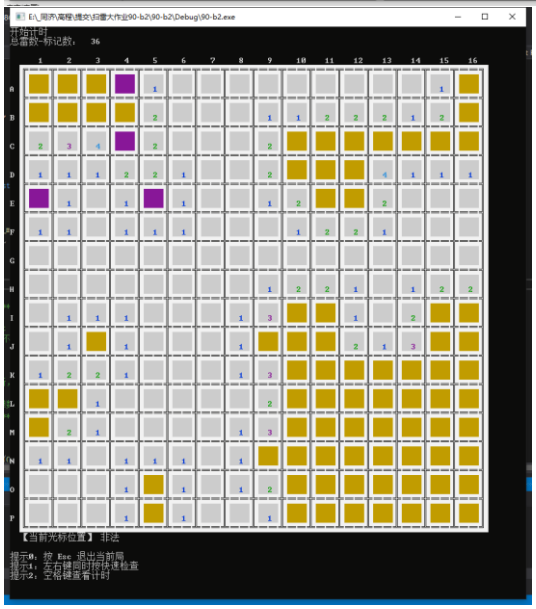
```
}

/*********************************************************************
功能:    以 *map 为 M[0][0]
         随机出一个含有 bombcnt 个炸弹，高 X 长 Y 的地图
         其中保证 M[X][Y] 肯定不是炸弹

注: 有效地图下标从 1 开始
         XY默认都为 1
         不进行参数合理性的判断，若参数不合理可能会出错
         炸弹处 M 值为 -1
         其它处 M 值为 周围炸弹数
*********************************************************************/
void newmap(int map[][Maxlen], const int Xlen, const int Ylen, int bombcnt, const int X, const int
Y)
{

    srand((unsigned int)(time(0)));

    for (int i = 0; i <= Xlen; i++)
        for (int j = 0; j <= Ylen; j++)
            map[i][j] = 0;

    for (int x, y, OK; bombcnt > 0; ) {
        x = rand() % Xlen + 1;
        y = rand() % Ylen + 1;

        OK = 1;
        for (int i = 0; i < 9 && OK; i++)
            OK = x + fx[i][0] != X && y + fx[i][1] != Y;

        if (!OK || map[x][y] < 0)
            continue;

        bombcnt--;
        map[x][y] = -1;
        for (int f = 0, xx, yy; f < 8; f++) {
            xx = x + fx[f][0];
            yy = y + fx[f][1];

            map[xx][yy] += map[xx][yy] >= 0;
        }
    }

    return;
#undef M
}
```

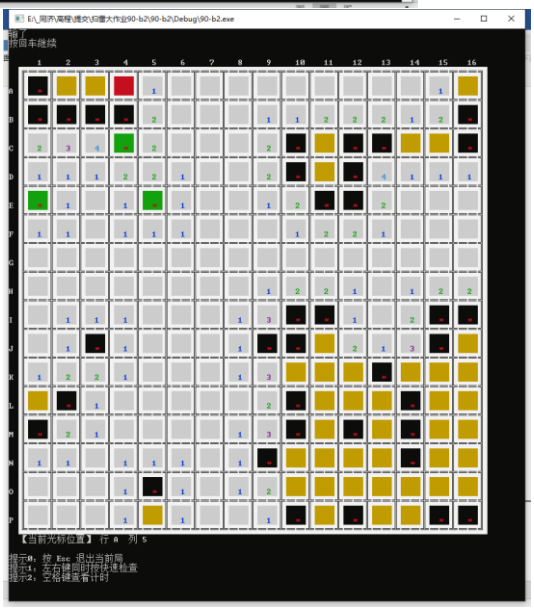
装 订 线

附上几张游戏界面截图









装 订 线