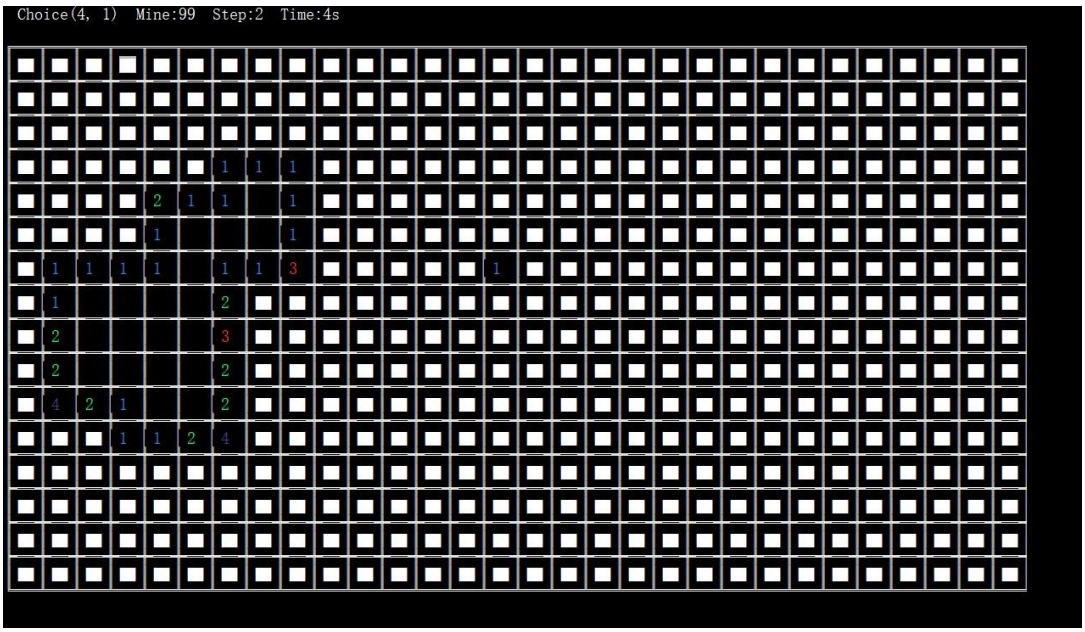
同濟大學

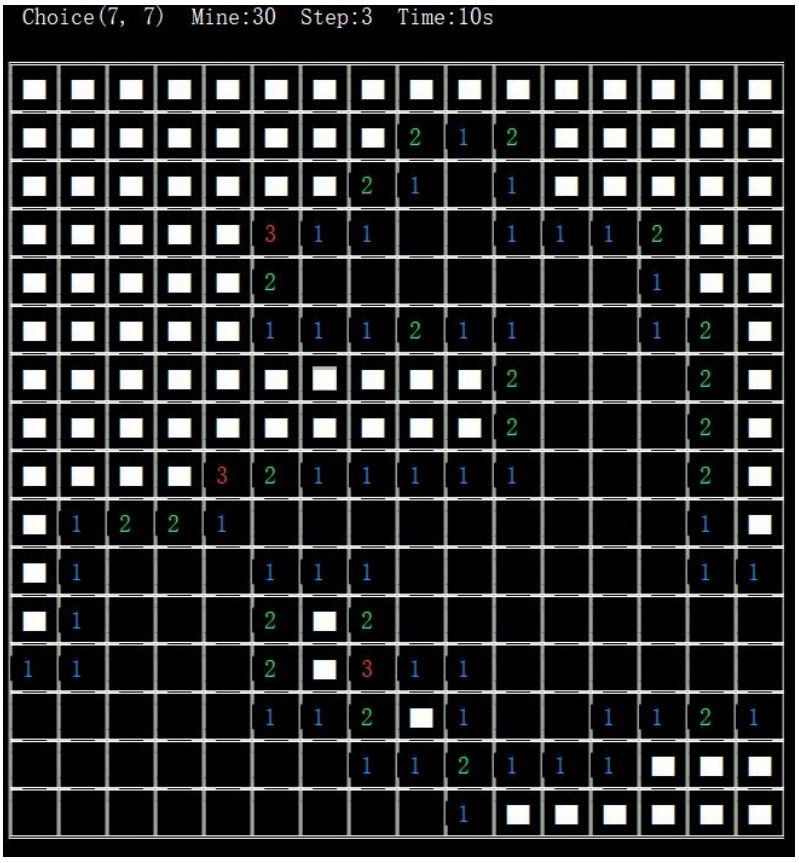
TONGJI UNIVERSITY

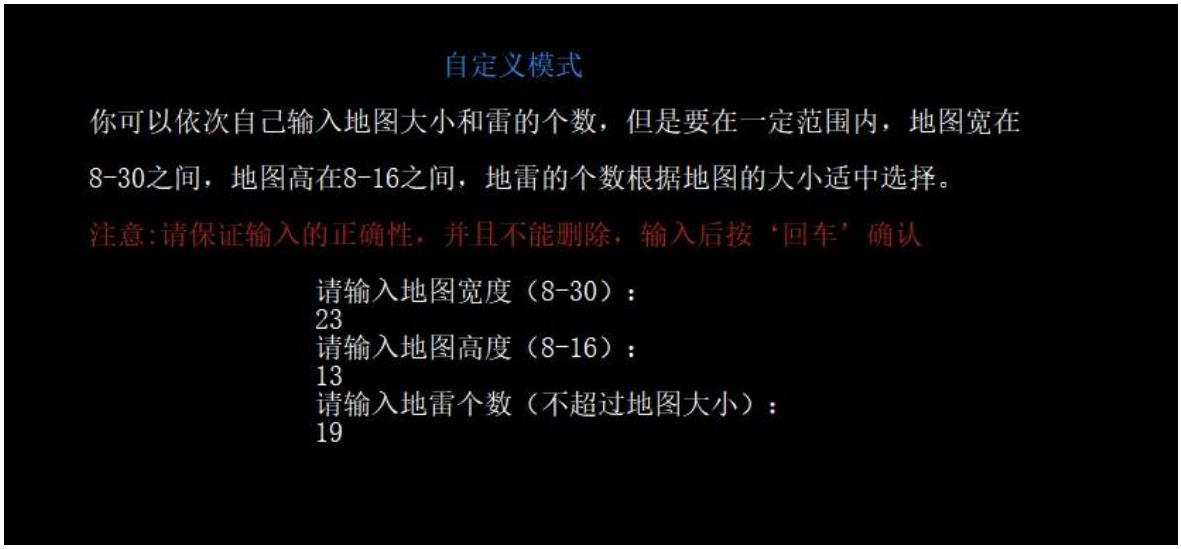
高程实验报告

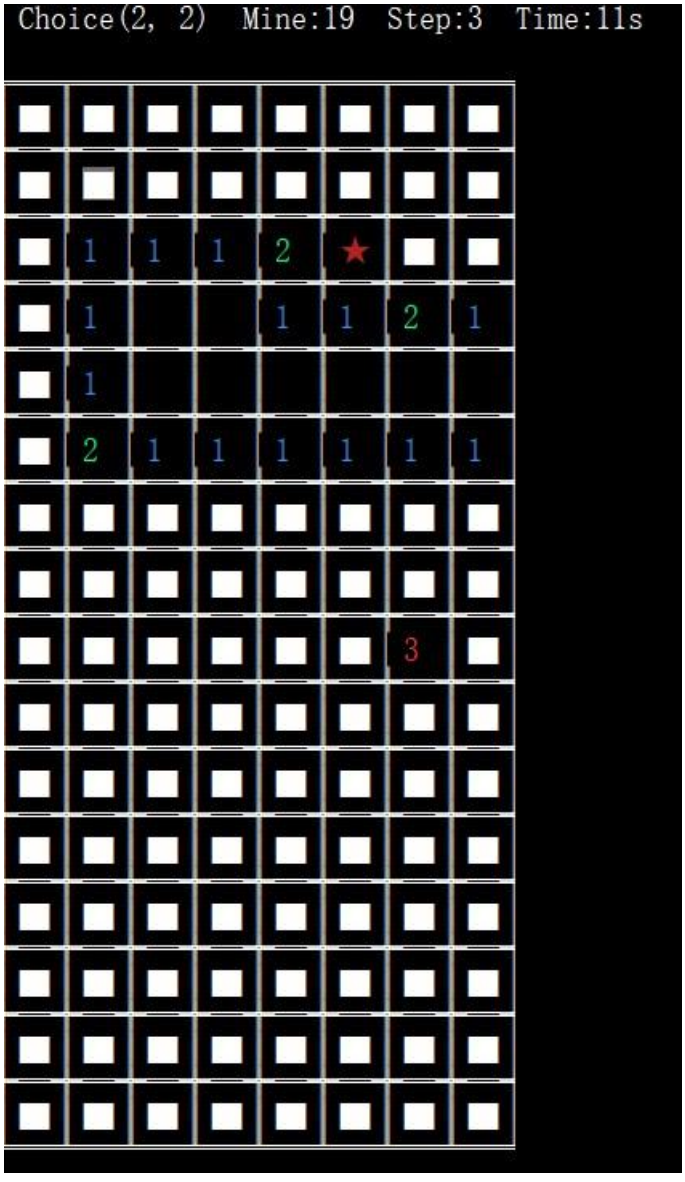
报告名称	控制台上的扫雷游戏
学院	新生院
班级	济勤学堂八班
学号	1952731
学生姓名	姚忠豪
指导教师	陈宇飞
完成日期	2019/12/20

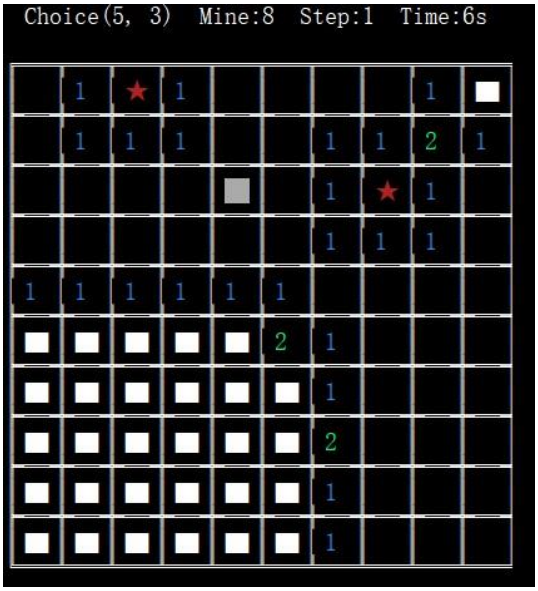
(一) 程序部分截图

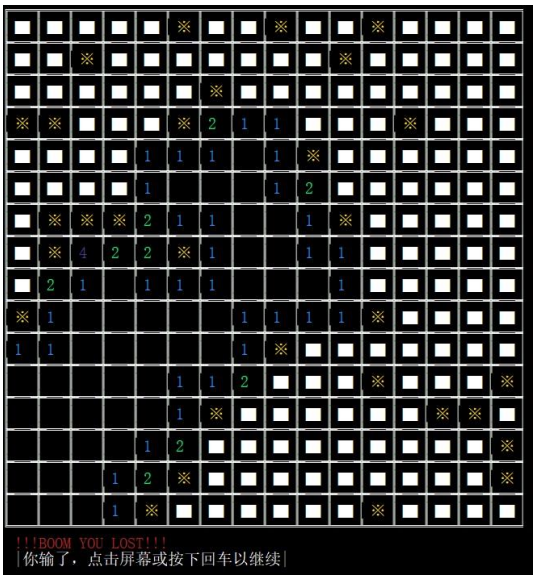

Choice(4, 1) Mine:99 Step:2 Time:4s


Choice(7, 7) Mine:30 Step:3 Time:10s


自定义模式
你可以依次自己输入地图大小和雷的个数，但是要在一定范围内，地图宽在
8-30之间，地图高在8-16之间，地雷的个数根据地图的大小适中选择。
注意:请保证输入的正确性，并且不能删除，输入后按‘回车’确认
请输入地图宽度（8-30）：
23
请输入地图高度（8-16）：
13
请输入地雷个数（不超过地图大小）：
19


Choice(2, 2) Mine:19 Step:3 Time:11s


Choice(5, 3) Mine:8 Step:1 Time:6s


!!!BOOM YOU LOST!!!
|你输了，点击屏幕或按下回车以继续|

（二）设计思路

(1)内部操作

可以联想到，扫雷游戏的内部实质上是一个地图数组，根据经典扫雷游戏规则，在游戏进行时，我们的“ 翻开地雷操作”，“ 标记地雷操作”，“地图扩散操作”，“判断输赢”等等，都是对地图数组进行的操作与遍历；所以这就需要使用各种函数（如 initMap 函数对地图数组元素初始化，翻开地块中的 loadingMap 函数对空白地块对应的数组元素进行拓展……），我认为这就是控制台扫雷游戏的成功的核心

(2)外部显示

外部显示主要依靠 **上善若水** 先生给的模板中 Graphic.cpp 进行绘制，其提供的 ClearScreen()，MovePos()，PutString() ，Update() 等等函数帮助我 完善了主要的绘制游戏界面功能

除此之外，显示功能也是将内部数组操作外显的桥梁（每一次操作后就绘制一遍地图，将自己的进行的操作更新到控制台以得到实时的操作结果）

(3)游戏化

关于扫雷的游戏化，主要从下面几个方面考虑：

游戏对用户的引导，提示。关于这一点可以使用菜单功能来实现，模板已经给出游戏菜单，另外，游戏说明也不能少（这样既可以给用户最佳的游戏体验，又可以防止用户操作失误而导致游戏瘫痪），所以需要利用 PutString()函数来绘制游戏说明和各种游戏提示；

用户对难度的选择。“一个只有一种难度的游戏不是好游戏”，当然这句话不是我说的，（其实这是加分项啊）（小声），关于初中高级和自定义的难度大小，可以通过改变地图数组的大小来实现的，所以只需要通过判断用户的选择来确定 地图宽高 雷的个数 等参数来实现；

游戏的可视化与 UI 设计。这就只是关于个人喜好和用户体验的角度了，通过 renderMap()函数中的图案进行个性化更改，比如标记符号

“★”，雷的符号“※”以及各种数字符号的颜色等等，此外，游戏相关的 UI 也需要设置完善，雷的个数，时间，以及已走步数等等，最后，输赢的判断也需要完善，游戏输了之后要显示所有雷的位置，赢了之后要显示所用的时间和步数；

（三）功能描述

此程序能实现的主要功能有：

(1)通过鼠标移动和点击或键盘的输入实现对控制台的操作

(2)实现基本的扫雷游戏功能，具备完整的 UI 元素（时间，步数，当前选择坐标，地雷数等)

(3)用户可选择困难程度（简单模式，一般模式，困难模式)，也可以自定义地图大小和地雷个数（在可控范围内）

(4)可以对鼠标或者键盘选中项进行渲染

（四）核心操作

(1)地图数组与游戏图标的对应

开始我理解为 0-8 绘制个数，9 绘制地雷，10 绘制标记，但是后来醒悟，忽略了扫雷的翻开操作。也就是说，第一次点击地块随机生成了一张地图，但是还未翻开的地块不能显示出来，而只有当其位置被点击或者被拓展到的时候才会显示，对此我选择的方法是**阶梯对应**，即随机生成的地图数组元素范围为 10-19（10-18 为非雷，19 为雷）此时对应的全是“未翻开”，其图标为“■”，颜色为白色；当某个地块被翻开后其对应的元素的值-10，范围为 0-9（0-8 代表的数字，9 代表雷)；相同的道理，当某块被标记时，其对应的元素的值+10（范围为 20-29)，被标记的图标为“★”，当其取消标记时，对应元素的值-10（范围回到 10-19)；

相关代码:

```
case 0:
    // 0的时候放置空白
    PutString("  ");
    break;
case 1:
    // 从1开始绘制数字
    PutStringWithColor(numMap, 30, 144, 255, 0, 0, 0);
    break;
case 2:
    PutStringWithColor(numMap, 0, 255, 127, 0, 0, 0);
    break;
case 3:
    PutStringWithColor(numMap, 255, 48, 48, 0, 0, 0);
    break;
case 4:
    PutStringWithColor(numMap, 72, 61, 139, 0, 0, 0);
    break;
case 5:
    PutStringWithColor(numMap, 255, 105, 180, 0, 0, 0);
    break;
case 6:
    PutStringWithColor(numMap, 148, 0, 211, 0, 0, 0);
    break;
case 7:
    PutStringWithColor(numMap, 139, 0, 0, 0, 0, 0);
    break;
case 8:
    PutStringWithColor(numMap, 139, 34, 82, 0, 0, 0);
    break;
case 9:
    // 9为地雷
    PutStringWithColor(" ※", 255, 215, 0, 0, 0, 0);
    break;
    //大于20为标记
case 20:
case 21:
case 22:
case 23:
case 24:
case 25:
case 26:
case 27:
case 28:
case 29:
    PutStringWithColor(" ★", 178, 34, 34, 0, 0, 0);
    //其余为未翻开
default:
```

```
PutString(" ■");
```

(2)随机埋雷与初始化地图数组

随机埋雷操作的要求为不在首落点埋雷（即第一次不会点中雷），对于随机埋雷的操作可以通过随机数生成然后再将随机数放缩到地图坐标大小，如果重复埋雷或者雷落在首落点上则无效，直到埋下指定数量的雷；

对于初始化地图，我采用的方法是对雷周围八邻域的非雷地块对于数组元素加 1，因为雷的数量相对于地图元素来说是少的，所以直接遍历雷能够减少遍历量；

相关代码：

```
// 随机埋雷
        srand((unsigned)time(NULL));
        int mineX, mineY;
        for (size_t i = 0; i < mineNum;) {
            mineX = rand() % mapHeight;
            mineY = rand() % mapWidth;
            if (mineX != pos.Y && mineY != pos.X && mapArray[mineX][mineY] != 19) {
                mapArray[mineX][mineY] = 19;
                i++;
            }

        }



// 确定数字
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                if (mapArray[i][j] >= 19) {
                    mapArrayPadding[i ][j ]++;
                    mapArrayPadding[i ][j+1]++;
                    mapArrayPadding[i ][j + 2]++;
                    mapArrayPadding[i+1][j ]++;
                    mapArrayPadding[i+1][j + 2]++;
                    mapArrayPadding[i + 2][j  ]++;
                    mapArrayPadding[i + 2][j+1]++;
                    mapArrayPadding[i + 2][j + 2]++;
                }
            }

        }
```

(3)对翻开的空白进行拓展

根据经典扫雷相关的规则，如果点中的地块周围没有雷（即为空白），

则会继续翻开其八邻域的地块，依此类推；

我使用的方法是函数递归实现，如果是空白，则调用该函数，翻开周围地块，如果再次遇到空白，则再次调用本函数，这样就实现了地图的拓展；

相关代码：

```
void loadingMap(size_t x,size_t y) {
        ……

        //首先翻开空白
        mapArray[x][y] = mapArray[x][y] - 10;

        //遍历赋值
        ……

        //拓展空白
        for (size_t i = x; i <= x + 2; i++) {
            for (size_t j = y; j <= y + 2; j++) {
                if (mapArrayPadding[i][j] >= 11 && mapArrayPadding[i][j] <= 18) {
                    mapArrayPadding[i][j] = mapArrayPadding[i][j] - 10;
                    for (size_t i = 0; i < mapHeight; i++) {
                        for (size_t j = 0; j < mapWidth; j++) {
                            mapArray[i][j] = mapArrayPadding[i + 1][j + 1];
                        }
                    }
                }
                else if (mapArrayPadding[i][j] == 10) {
                    loadingMap(i - 1, j - 1); //递归调用
                }
            }
        }


    }
```

(4)判断输赢

赢：赢的判断主要是对地图数组的元素进行遍历如果符合胜利的条件，立刻改变 bool 判断值，调用函数显示胜利提示；

输：点中雷，立刻改变 bool 判断值，调用函数显示失败提示；

相关代码：

#胜利：

```
void judgeWin() {
        //判断是否胜利
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                if (mapArray[i][j] >= 9 && mapArray[i][j] <= 28) {
                    isWin = false;
                }
            }
        }

    }
```

……

调用方式：

```
//判断是否胜利
            if (operation ==1|| operation==2) {
                judgeWin();
                if (isWin) {
                    renderWinTip();//绘制成功提示
                    ……
                    // 等待输入
                    ……

                    //游戏结束
                    gameFlag = false;
                }

            }
```

#失败：

```
// 如果翻开的是雷，则失败
        if (mapArray[x][y] == 19) {
            isLose = true;

        }


//判断是否失败
            if (isLose) {
                renderLoseTip();//绘制失败提示
                    ……
                    // 等待输入
                    ……
                //游戏结束
                gameFlag = false;

            }
```

（五）遇到的问题和解决办法

(1)问题 1：代码模板对于我这种小白来说真的好难好难看懂啊

解决办法：

最笨的解决办法：多看!!!!

不要求把每个函数的实质看懂，只要熟悉他的功能，调用方式与调用后的结果就好了；

我花了整整两个晚上才大概看懂了各个函数的基本功能 QAQ，熟悉了每个函数之后，再查看函数的所有引用，记住它们在哪里被调用了；

如果这样再不清楚，那就更改相关函数的内容，再编译运行，观察与原函数显示有何不同，依次了解其明确的功能

学有余力之后，再去查阅相关文献资料深入了解各个 cpp 中各函数的实质与各种没见过的数据类型

(2)问题 2：调用绘制游戏说明函数后直接闪回菜单

解决方法：

分析原因，如果完善的函数 RenderIntro() 中只有清屏并绘制游戏说明的功能，那么则会直接返回菜单，于是我思考能不能用一个语句等待用户返回菜单，发现在 GameEngine 中的 checkChoice 函数，用于检查输入和操作（即获得鼠标和键盘的状态），于是就可以利用此函数以等待用户操作以返回菜单；

之后的显示胜利或失败提示也用到了该方法

相关代码：

```
//等待输入
        while (true) {
            checkChoice();
            if (operation == 1 || operation == 2) break;

        }
```

(3)问题 3：地图数组处理八邻域时的边界处理问题

解决办法:

对地图数组 mapArray[mapWidth][mapHeight]进行填充，行列数+1，利用 mapArrayPadding[mapWidth+1][mapHeight+1]处理边界值，于是每次需要对地图数组的边界进行处理或者判断时，都需要 mapArrayPadding 数组进行辅助，并保证外围填充的值不影响原地图数组元素;

相关代码:

//初始化地图时需要处理边界，于是利用了 mapArrayPadding 数组

```
for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                mapArrayPadding[i + 1][j + 1] = mapArray[i][j];
            }
        }

        // 确定数字
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                if (mapArray[i][j] >= 19) {
                    mapArrayPadding[i ][j ]++;
                    mapArrayPadding[i ][j+1]++;
                    mapArrayPadding[i ][j + 2]++;
                    mapArrayPadding[i+1][j ]++;
                    mapArrayPadding[i+1][j + 2]++;
                    mapArrayPadding[i + 2][j  ]++;
                    mapArrayPadding[i + 2][j+1]++;
                    mapArrayPadding[i + 2][j + 2]++;
                }
            }
        }

        // 更新雷的值
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                mapArray[i][j] = mapArrayPadding[i + 1][j + 1];
                if (mapArray[i][j] > 19) mapArray[i][j] = 19;
            }

        }
```

(4)问题 4：困难程度选择时对数组大小的定义问题

解决办法：

因为静态变量对数组的大小定义只能用常数，也就是不能改变的数，如果根据用户的不同选择或者自定义来确定地图的大小，则数组的大小必定是可以改变的值，于是我运用了动态数组来实现，也就是每次调用一次 Play()函数就 new 开辟一段连续的内存，调用 DestroyGame()函数 delete 开辟的内存；

但是又遇到一个新的问题，控制台绘制出来的地图与推理的不符，经过不断注释，不断调试，最终发现了问题所在：

memcpy_s(mapCanvas, mapWidth* mapHeight, mapArray, mapWidth* mapHeight);

以上是原来 mapArray 数组不是动态数组时的 updateMap 函数内容 memcpy_s()函数的功能是拷贝指定大小的内存

当把原来的内容换成遍历赋值的话，则运行良好，地图绘制正确

```
void updateMap() {
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                mapCanvas[j + i * mapWidth] = mapArray[i][j];
            }
        }

    }
```

(5)问题 5：用户自定义地图大小过程中的输入问题

解决办法：

由于惯性思维，以为解决了模式选择就可以解决自定义（直接 cin 输入就是了!），但是本项目模板并没有 include iostream，而且本项目所有在控制台显示的东西都是靠绘制 char 或者 string 来实现的，以我（小白视角）的理解，应该不能直接 cout cin getchar scanf printf 等等。于是我使用了_getch()函数，通过键入的数字接受字符通过 putChar 来绘制到画板上，再获得用户输入的数

相关代码：

//以输入地图宽度为例

```
//获得地图宽度
        MovePos(46, 10);
        PutString("请输入地图宽度（8-30）：");
```

```
    Update();
    MovePos(46, 11);
    while (ch != 13) {
        n = n * 10 + ch - '0';
        while (true) {
            ch = _getch(); if (ch >= '0' && ch <= '9' || ch == 13) break;
        }
        PutChar(ch);
        Update();
        i++;
        MovePos(46+i, 11);

    }
```

（六）心得体会

（1）大开眼界

这是我深入接触到的第一个比较大的工程文件，里面引入了一些从来没有见过的库；emmm 按我的理解，本次大作业的目的就是让我们了解一个完整的项目先需要具备那些东西，然后再去完善些什么东西，也就是说，这能够提醒我们在今后漫长的打代码的过程中，如何去高效地，精确地写一个项目；就拿这个控制台扫雷游戏来说，TA 给的框架已经具备了大部分的基本功能（指引入画板和鼠标操作等等），然后我们的任务就是去完善相关算法，完善内部操作，完善游戏化等等；所以我认为，我在以后的写代码的过程中，也应该按照这个流程，先把基本功能写好写细，这样在完善的过程中压力就会相对小一些；

（2）扫雷算法

其实主要是指数组的操作，首先我是玩了好多局扫雷才琢磨出如何去进行数组操作的，其实算法并没有标准答案，只有优化程度或者是安全程度之分，我们要注重效率，注重安全……（当然，能力有限，并没有琢磨出十分完美的算法，可能有些算法在大家眼里是非常傻的那种嘤嘤嘤 QAQ）

（3）感谢大作业

？？？？为什么还要感谢大作业！我是不是疯了？？？

不是，纯粹是因为我做完大作业会有一点点成就感 hh，第一个自己写了核心算法的游戏必须 mark 一下，之前我只是会点击鼠标会敲键盘的

一个学生，现在我可以说这个游戏是我完成的。

当然我还可以把游戏分享给别人，分享给我弟弟（虽然他说这个游戏好简陋啊，但是我知道他还是在默默崇拜我的），分享给同学，学长...（不说了，贴图叭）

（4）吐槽

（为什么一定要吐槽？？我吐槽我自己行吗，我太菜了 QAQ）

如果真的要吐槽的话，那就吐槽我还没见过的 TA 们，你们什么时候出现啊啊啊，最后一节课一定要来!!!!

如果真的真的要吐槽的话，还有一个就是，为什么我有时候点击屏幕会一闪一闪的，我也没有一直清屏啊

（5）完结撒花

本学期的高程课接近尾声了，学到许多(滑稽)，真的十分感谢陈大哥和 TA 们，还有所有高程班的同学们，你们真的太可爱了!!!!!!!

（七）源代码（相比于框架修改或增添的函数）

```
/****************************************
    Function:  renderMap()
    Parameter: None(void)
    Return:    None(void)
    Description:
    绘制当前地图
    ****************************************/
    void renderMap() {

        // 从行开始遍历
        for (size_t i = 0; i < mapHeight + 1; i++) {
            // 移动坐标至行首 并绘制行首的字符
            MovePos(0, (SHORT)i * 2 + 3);
            if (i == 0) {
                PutString("╔");
            }
            else if (i == mapHeight) {
                PutString("╚");
            }
            else {
                PutString("╠");
            }

            // 遍历列，绘制边界
            for (size_t j = 0; j < mapWidth; j++) {
                // 移动至确切的坐标，绘制方格的边界
                MovePos(2 + (SHORT)j * 8, (SHORT)i * 2 + 3);
                if (i == 0) {
                    if (j < mapWidth - 1) {
                        PutString("════════╦");
                    }
                    else {
                        PutString("════════╗");
                    }
                }
                else if (i == mapHeight) {
                    if (j < mapWidth - 1) {
                        PutString("════════╩");
                    }
                    else {
                        PutString("════════╝");
                    }
                }
                else {
                    if (j < mapWidth - 1) {
                        PutString("════════╬");
                    }
```

```
                    else {
                        PutString("═══╣ ");
                    }
                }
            }

            // 绘制地图
            if (i > 0 && i < mapHeight + 1) {
                // 移动至行首，绘制边界字符
                MovePos(0, (SHORT)i * 2 + 2);
                PutString("╟ ");
                // 遍历列 绘制地雷
                for (size_t j = 0; j < mapWidth; j++) {
                    MovePos(2 + (SHORT)j * 5, (SHORT)i * 2 + 2);    // 移动至确切坐标
                    const size_t mapIndex = (i - 1) * mapWidth + j; // 确定地图数组的下标
                    char numMap[8] = "   ";                          // 确定数字字符串
                    numMap[1] = '0' + mapCanvas[mapIndex];          // 当
mapCanvas[mapIndex]为1到8时，将其转换成字符串
                    switch (mapCanvas[mapIndex]) {
                    case 0:
                        // 0的时候放置空白
                        PutString("   ");
                        break;
                    case 1:
                        // 从1开始绘制数字
                        PutStringWithColor(numMap, 30, 144, 255, 0, 0, 0);
                        break;
                    case 2:
                        PutStringWithColor(numMap, 0, 255, 127, 0, 0, 0);
                        break;
                    case 3:
                        PutStringWithColor(numMap, 255, 48, 48, 0, 0, 0);
                        break;
                    case 4:
                        PutStringWithColor(numMap, 72, 61, 139, 0, 0, 0);
                        break;
                    case 5:
                        PutStringWithColor(numMap, 255, 105, 180, 0, 0, 0);
                        break;
                    case 6:
                        PutStringWithColor(numMap, 148, 0, 211, 0, 0, 0);
                        break;
                    case 7:
                        PutStringWithColor(numMap, 139, 0, 0, 0, 0, 0);
                        break;
                    case 8:
                        PutStringWithColor(numMap, 139, 34, 82, 0, 0, 0);
                        break;
                    case 9:
                        // 9为地雷
                        PutStringWithColor(" ※", 255, 215, 0, 0, 0, 0);
                        break;
                        //大于20为标记
                    case 20:
```

```
                case 21:
                case 22:
                case 23:
                case 24:
                case 25:
                case 26:
                case 27:
                case 28:
                case 29:
                    PutStringWithColor("★", 178, 34, 34, 0, 0, 0);
                    //其余为未翻开
                default:
                    PutString("■");
                }

                MovePos(5 + (SHORT)j * 5, (SHORT)i * 2 + 2);
                PutString("|");
            }
        }
    }

    // 将地图更新到屏幕
    Update();
}

/*****************************************
Function:  updateMap()
Parameter: None(void)
Return:    None(void)
Description:
将二维数组的数据复制到一维数组中
*****************************************/
void updateMap() {
    for (size_t i = 0; i < mapHeight; i++) {
        for (size_t j = 0; j < mapWidth; j++) {
            mapCanvas[j + i * mapWidth] = mapArray[i][j];
        }
    }
}

/*****************************************
Function:  patternCust()
Parameter: None(void)
Return:    None(void)
Description:
得到用户自定义地图的大小与雷的个数
*****************************************/
void patternCust() {
    //相关变量
    char ch='0';
    size_t n=0;
    size_t i = 0;

    //绘制说明
```

```
        ClearScreen();                                  //清屏
        MovePos(55, 2);
        PutStringWithColor("自定义模式", 30, 144, 255, 0, 0, 0);
        MovePos(30, 4);
        PutString("你可以依次自己输入地图大小和雷的个数，但是要在一定范围内，地图宽在");
        MovePos(30, 6);
        PutString("8-30之间，地图高在8-16之间，地雷的个数根据地图的大小适中选择。");
        MovePos(30, 8);
        PutStringWithColor("注意:请保证输入的正确性，并且不能删除，输入后按‘回车’确认",
178, 34, 34, 0, 0, 0);
        Update();


        //获得地图宽度
        MovePos(46, 10);
        PutString("请输入地图宽度（8-30）：");
        Update();
        MovePos(46, 11);
        while (ch != 13) {
            n = n * 10 + ch - '0';
            while (true) {
                ch = _getch(); if (ch >= '0' && ch <= '9' || ch == 13) break;
            }
            PutChar(ch);
            Update();
            i++;
            MovePos(46+i, 11);
        }
        mapWidth = n;
        ch = '0'; n = 0; i = 0;

        //获得地图高度
        MovePos(46, 12);
        PutString("请输入地图高度（8-16）：");
        Update();
        MovePos(46, 13);
        while (ch != 13) {
            n = n * 10 + ch - '0';
            while (true) {
                ch = _getch(); if (ch >= '0' && ch <= '9' || ch == 13) break;
            }
            PutChar(ch);
            Update();
            i++;
            MovePos(46 + i, 13);
        }
        mapHeight = n;
        ch = '0'; n = 0; i = 0;

        //获得地雷个数
        MovePos(46, 14);
        PutString("请输入地雷个数（不超过地图大小）：");
        Update();
        MovePos(46, 15);
```

```
    while (ch != 13) {
        n = n * 10 + ch - '0';
        while (true) {
            ch = _getch(); if (ch >= '0' && ch <= '9' || ch == 13) break;
        }
        PutChar(ch);
        Update();
        i++;
        MovePos(46 + i, 15);
    }
    mineNum = n;

}


/*****************************************
Function:  patternChoice()
Parameter: None(void)
Return:    None(void)
Description:
显示模式选择，根据选择的困难程度确定地图的大小与雷的个数
*****************************************/
void patternChoice() {

    //绘制说明
    ClearScreen();//清屏
    MovePos(62, 2);
    PutStringWithColor("困难程度", 30, 144, 255, 0, 0, 0);
    MovePos(38, 4);
    PutString("|简单模式|    地图大小：10x10 雷：10       (按‘1’)");
    MovePos(38, 6);
    PutString("|一般模式|    地图大小：16x16 雷：30       (按‘2’)");
    MovePos(38, 8);
    PutString("|困难模式|    地图大小：16x30 雷：99       (按‘3’)");
    MovePos(38, 10);
    PutString("|自定义模式|  地图宽：8-30 地图高：8-16    (按‘4’)");
    Update();

    //检查输入
    char choice;
    while (true) {
        choice = _getch();
        if (choice == '1' || choice == '2' || choice == '3'||choice=='4') break;
    }

    //根据输入确定困难程度
    switch (choice) {
    case '1':
        //简单模式
        mapWidth = 10;
        mapHeight = 10;
        mineNum = 10;
        break;
    case '2':
```

```
            //一般模式
            mapWidth = 16;
            mapHeight = 16;
            mineNum = 30;
            break;
        case '3':
            //困难模式
            mapWidth = 30;
            mapHeight = 16;
            mineNum = 99;
            break;
        case '4':
            //自定义难度
            patternCust();
        }
        ClearScreen();//清屏
    }

    /**************************************
    Function:  InitGame()
    Parameter: None(void)
    Return:    None(void)
    Description:
    初始化游戏
    **************************************/
    void InitGame() {

        // 清屏
        ClearScreen();

        // 确定困难程度
        patternChoice();

        // 分配地图数组空间
        mapArray = new UCHAR * [mapHeight];
        mapArray[0] = new UCHAR[mapWidth * mapHeight];
        for (size_t i = 1; i < mapHeight; i++) mapArray[i] = mapArray[i - 1] + mapWidth;
        mapArrayPadding = new UCHAR * [mapHeight+2];
        mapArrayPadding[0] = new UCHAR[(mapWidth+2) * (mapHeight+2)];
        for (size_t i = 1; i < mapHeight+2; i++) mapArrayPadding[i] = mapArrayPadding[i -
1] + mapWidth+2;
        mapCanvas = new UCHAR[mapWidth * mapHeight];

        // 初始化地图数组
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                mapArray[i][j] = 10;
            }
        }

        // 更新并绘制地图
        updateMap();
        renderMap();
```

```
    }

/****************************************
    Fuction: loadingMap()
    Parameter:
    Return:
    Description:
    翻开地块并且利用递归做延伸拓展
    ****************************************/
    void loadingMap(size_t x,size_t y) {

        //首先翻开空白
        mapArray[x][y] = mapArray[x][y] - 10;

        //遍历赋值
        for (size_t i = 0; i < mapHeight+2; i++) {
            for (size_t j = 0; j < mapWidth+2; j++) {
                mapArrayPadding[i][j] = 100;
            }
        }
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                mapArrayPadding[i + 1][j + 1] = mapArray[i][j];
            }
        }

        //拓展空白
        for (size_t i = x; i <= x + 2; i++) {
            for (size_t j = y; j <= y + 2; j++) {
                if (mapArrayPadding[i][j] >= 11 && mapArrayPadding[i][j] <= 18) {
                    mapArrayPadding[i][j] = mapArrayPadding[i][j] - 10;
                    for (size_t i = 0; i < mapHeight; i++) {
                        for (size_t j = 0; j < mapWidth; j++) {
                            mapArray[i][j] = mapArrayPadding[i + 1][j + 1];
                        }
                    }
                }
                else if (mapArrayPadding[i][j] == 10) {
                    loadingMap(i - 1, j - 1); //递归操作
                }
            }
        }

    }

    /****************************************
    Function:  renderFailTip
    Parameter: None(void)
    Return:    None(void)
    Description:
    显示失败提示
    ****************************************/
    void renderLoseTip() {
```

```
    //显示雷
    for (size_t i = 0; i < mapHeight; i++) {
        for (size_t j = 0; j < mapWidth; j++) {
            if (mapArray[i][j] == 19|| mapArray[i][j] == 29) mapArray[i][j] = 9;
        }
    }

    // 更新并绘制地图
    updateMap();
    renderMap();

    //显示失败提示
    MovePos(2,2*mapHeight+3);
    PutStringWithColor("!!!BOOM YOU LOST!!!", 178, 34, 34, 0, 0, 0);
    MovePos(2, 2 * mapHeight + 4);
    PutString("|你输了，点击屏幕或按下回车以继续|");
    Update();
}

/****************************************
Function:  renderWinTip
Parameter: None(void)
Return:    None(void)
Description:
显示成功提示
****************************************/
void renderWinTip() {

    //显示步数和所用时间
    MovePos(12, 1);
    char strStep[32] = "";
    sprintf_s(strStep, "Step:%u   ", stepNum);
    PutString(strStep);
    char strTime[32] = "";
    sprintf_s(strTime, "Time:%ds  ", (int)(timeEnd - timeStart) / CLOCKS_PER_SEC);
    PutStringWithColor(strTime, 30, 144, 255, 0, 0, 0);

    //显示成功提示
    MovePos(2, 2 * mapHeight + 3);
    PutStringWithColor("!!!! YOU  WIN !!!!", 30, 144, 255, 0, 0, 0);
    MovePos(2, 2 * mapHeight + 4);
    PutString("|你赢了，点击屏幕或按下回车继续|");
    Update();
}

/****************************************
Function:  initMap(COORD pos)
Parameter: COORD pos
Return:    None(void)
Description:
埋雷操作，初始化地图
****************************************/
void initMap(COORD pos) {
```

```
    // 随机埋雷
    srand((unsigned)time(NULL));
    int mineX, mineY;
    for (size_t i = 0; i < mineNum;) {
        mineX = rand() % mapHeight;
        mineY = rand() % mapWidth;
        if (mineX != pos.Y && mineY != pos.X && mapArray[mineX][mineY] != 19) {
            mapArray[mineX][mineY] = 19;
            i++;
        }
    }

    //遍历赋值
    for (size_t i = 0; i < mapHeight; i++) {
        for (size_t j = 0; j < mapWidth; j++) {
            mapArrayPadding[i + 1][j + 1] = mapArray[i][j];
        }
    }

    // 确定数字
    for (size_t i = 0; i < mapHeight; i++) {
        for (size_t j = 0; j < mapWidth; j++) {
            if (mapArray[i][j] >= 19) {
                mapArrayPadding[i ][j ]++;
                mapArrayPadding[i ][j+1]++;
                mapArrayPadding[i ][j + 2]++;
                mapArrayPadding[i+1][j ]++;
                mapArrayPadding[i+1][j + 2]++;
                mapArrayPadding[i + 2][j  ]++;
                mapArrayPadding[i + 2][j+1]++;
                mapArrayPadding[i + 2][j + 2]++;
            }
        }
    }

    // 更新雷的值
    for (size_t i = 0; i < mapHeight; i++) {
        for (size_t j = 0; j < mapWidth; j++) {
            mapArray[i][j] = mapArrayPadding[i + 1][j + 1];
            if (mapArray[i][j] > 19) mapArray[i][j] = 19;
        }
    }

    // 更新并绘制地图
    updateMap();
    renderMap();
}

/**************************************
Function:  digBlock()
Parameter: COORD(pos)
Return:    None(void)
Description:
翻开地块，并判断是否为雷
```

```
****************************************/
void digBlock(COORD pos) {

    if (isFirst) {
        // 如果是第一步走，则先初始化地图
        initMap(pos);
        isFirst = false; // 将第一步设置为否
    }

    // 获取翻开的地块坐标
    size_t x = pos.Y;
    size_t y = pos.X;

    // 如果翻开的是雷，则失败
    if (mapArray[x][y] == 19) {
        isLose = true;
    }

    // 如果是数字，直接翻开
    else if (mapArray[x][y] >= 11 && mapArray[x][y] <= 18) {
        mapArray[x][y] = mapArray[x][y] - 10;
        stepNum++;
        // 更新并绘制地图
        updateMap();
        renderMap();
    }

    // 如果是空白，则扩散
    else if (mapArray[x][y] == 10) {
        loadingMap(x, y);
        stepNum++;
        // 更新并绘制地图
        updateMap();
        renderMap();
    }
}

/****************************************
Function:  flagBlock(COORD pos)
Parameter: COORD pos
Return:    None(void)
Description:
标记或清除标记
****************************************/
void flagBlock(COORD pos) {

    // 标记
    if (mapArray[pos.Y][pos.X] >= 10 && mapArray[pos.Y][pos.X] <= 19) {
        mapArray[pos.Y][pos.X] = mapArray[pos.Y][pos.X] + 10;
        flagNum++;
    }

    // 清除标记
    else if(mapArray[pos.Y][pos.X] >= 20 && mapArray[pos.Y][pos.X] <= 29) {
```

```
            mapArray[pos.Y][pos.X] = mapArray[pos.Y][pos.X] - 10;
            flagNum--;
        }

        // 更新并绘制地图
        updateMap();
        renderMap();
    }

    /**************************************
    Function:  judgeWin()
    Parameter: None(void)
    Return:    None(void)
    Description:
    判断是否成功
    **************************************/
    void judgeWin() {
        //判断是否胜利
        for (size_t i = 0; i < mapHeight; i++) {
            for (size_t j = 0; j < mapWidth; j++) {
                if (mapArray[i][j] >= 9 && mapArray[i][j] <= 28) {
                    isWin = false;
                }
            }
        }

    }


/**************************************
    Function:  Play()
    Parameter: None(void)
    Return:    None(void)
    Description:
    开始游戏
    **************************************/
    void Play() {
        timeStart = clock();                                    //开始计时
        flagNum = 0;                                            //初始化标记数
        stepNum = 0;                                            //初始化步数
        isFirst = true;
        isLose = false;
        gameFlag = true;

        //进入循环
        while (gameFlag) {
            timeEnd = clock();
            isWin = true;
            checkChoice(); // 检查输入

            // 查看当前坐标是否需要更新背景
            if (posChoice.X != posChoiceOld.X || posChoice.Y != posChoiceOld.Y) {
                clearChoiceBackground(posChoiceOld);
                posChoiceOld = posChoice;
```

```
}
renderChoiceBackground(posChoice);

// 放置当前选择位置的字符串：‘选择位置’，‘步数’，‘剩余地雷数’，‘时间’
MovePos(2, 0);
char strChoice[32] = "";
sprintf_s(strChoice, "Choice(%u, %u)  ", posChoice.X+1, posChoice.Y+1);
PutString(strChoice);
char strFlag[32] = "";
sprintf_s(strFlag, "Mine:%d  ", mineNum-flagNum);
PutString(strFlag);
char strStep[32] = "";
sprintf_s(strStep, "Step:%u  ", stepNum);
PutString(strStep);
char strTime[32] = "";
sprintf_s(strTime, "Time:%ds  ", (int)(timeEnd-timeStart)/CLOCKS_PER_SEC);
PutString(strTime);
Update();

// 根据鼠标或键盘行为执行相应操作
switch (operation) {
case 1:
    // 翻块
    digBlock(posChoice);
    break;
case 2:
    // 标记
    flagBlock(posChoice);
    break;
}

//判断是否失败
if (isLose) {
    renderLoseTip();//绘制失败提示
    // 等待输入
    while (true) {
        checkChoice();
        if (operation == 1 || operation == 2) break;
    }
    //游戏结束
    gameFlag = false;
}

//判断是否胜利
if (operation ==1|| operation==2) {
    judgeWin();
    if (isWin) {
        renderWinTip();//绘制成功提示
        // 等待输入
        while (true) {
            checkChoice();
            if (operation == 1 || operation == 2) break;
        }
        //游戏结束
```

```
                    gameFlag = false;
                }
            }

            //处理每帧的事务
            updateMap(); // 更新地图画板
            Update();    // 更新操作到屏幕

            frame++;  // 渲染帧数自增
            clock_t elapsed = 25 - (clock() - tic); // 检查上一帧渲染时间，并计算与25的差值
            Sleep(elapsed > 0 ? elapsed : 0);       // 若差值大于零，则休眠该差值的毫秒数，
以确保每帧渲染不超过50帧
            tic = clock();                          // 更新上一次记录的时间
        }

    }
```