同济大学

Tongji University

《高级语言程序设计》

实验报告

报告名称：扫雷游戏的设计与实现

班　　级：软件工程

学　　号：2253744

姓　　名：林觉凯

完成日期：2023年12月24日

1.题目描述

这次的扫雷大作业的每一个选项都是循序渐进的，大致可以分为两个部分：内部数组部分(1-4)和伪图形界面部分(5-9)，要求我们用不同的方式将扫雷这一游戏完整地呈现出来。

内部数组部分(1-4)：

1.选择难度并显示内部数组。一共有三个难度(1-3)，选择难度选项后随机在雷区生成相应数目的雷数，并且计算每一个非雷位置周边的雷数，最后打印出来(类似与先前的小作业)。

2.输入初始位置并显示被打开的初始区域。同样一共有三个难度(1-3)，键盘输入行列坐标(严格区分大小写，行号必须大写)，这里需要注意要保证输入的第一个位置非雷且周围的八个位置均非雷。(保证第一次按键就遇到雷游戏就结束)。同时要有“扩散”的功能，即找到所有相邻的0，并向四周扩散，扩散的终止条件为到达边界或者到达所有非0但不是雷的位置。第二题的打印初始区域也和第一题不一样，需要我们采用亮色反显的方式，达到醒目的效果。其余位置(没有被打开)仍为“X”。

3.内部数组基础版。和2的要求相同，打开初始区域，但是可以循环进行游戏，即继续输入行列坐标，同样也有相对应的打开操作，如果遇到雷就显示游戏结束。循环进行游戏，直到找到所有雷(游戏成功)或者输入的坐标为雷的坐标(游戏失败)。

4.内部数组完整版。和3的要求大致相同，只是新增了几个输入。即可以用“&”显示本局游戏已运行的时间、“！行列”为标记该点为雷、“#行列”为取消该点标记为雷。循环进行游戏，直到找到所有雷(游戏成功)或者输入的坐标为雷的坐标(游戏失败)。

伪图形界面部分(1-4)：

5.画出伪图形化的框架并显示内部数据。和1选项类似，生成雷并且计算非雷位置周围的雷数。加上画出相应难度的伪图形的边框框架，将内部数组显示(其中0不显示)。

6.在伪图形化的框架上移动鼠标，判断鼠标的位置。在子题目5的基础上在框架上移动鼠标，并且显示当前鼠标的行列位置(移动到框架外或者网格线上则非法)。

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。子题目2和6的结合版。

8.伪图形化游戏的基础版。同子题目7，打开初始区域。继续游戏，鼠标左键单击表示选择非雷位置，要求同子题目2；鼠标右键标记某处为雷。键盘ESC退出游戏。循环进行游戏，直到找到所有雷(游戏成功)或者左键单击为雷的坐标(游戏失败)。

9.伪图形化游戏完整版。在子题目8的基础上，显示剩余雷数、计算本局游戏已运行时间(空格显示时间、游戏结束时显示时间)。

2.整体设计思路

2.1.程序的主体架构设计

一开始分析题目的时候我发现每一个选项都是有三个难度选择，八个功能选项，当时我想着要不要直接开一个最大的二维数组直接将三种难度包含了得了，最后我考虑到每一个函数都不宜太长，而且如果直接开一个最大的数组，每一个函数里面还要进行判断，这样会使每一个函数太长而且判断太多容易出错，因此我最后决定依据难度将这3*8=24个要实现的部分分为三个大部分，即我在每一个函

数名后面都有加了_1,_2和_3符号，表示难度1、难度2、难度3三种不同的难度，并且在每一个函数里面分别选择相应的选项参数(choice)，完成每一个选项的该难度实现。_1,_2和_3之间的代码可以实现复用，题目类似，只要改一些参数即可，使得程序的主题框架一目了然且也不麻烦。

这是我对于三个难度的处理方式，依据题目我还得出整体的题目分为内部数组实现部分和伪图形界面实现部分，因此，我在main函数里将它们先分为了两大部分：内部数组实现部分和伪图形界面实现部分，这里我就构造了两个相应cpp中的主函数：Inner_difficulty_x和Graph_difficulty_x，这两个函数分别是mine_sweeper_base.cpp和mine_sweeper_graph.cpp中的“主函数”(x代表难度选择)。这两个函数中有相应的选项判断，再在这两个函数里面调用mine_sweeper_base.cpp或mine_sweeper_graph.cpp中的其他解决函数来达到实现功能的目的。

通过以上两个划分依据，程序的大致框架就能显示出来了。

以子题目8和子题目9为准，以下是我的程序实现思路：首先进入主菜单，选择相应的功能选项，(比如这里选择了8或者9选项)，接下来进入副菜单，即选择相应的难度，通过这两个选择程序跳到相应的Inner_difficulty_x或者是Graph_difficulty_x里，这里以子题目8和9为准，跳到Graph_difficulty_x里。然后进入Graph_difficulty_x中，8和9的共有操作首先为打印初始界面，调用相应的打印初始界面函数，然后调用Graph_game_x(这个函数里面也有调用别的函数，比如鼠标操作函数、判断输赢函数、扩散雷区函数等等)，总之就是多个函数的调用进行相应的功能游戏。9比8多打印剩余雷数和显示游戏运行时间，我们只需在每次输出时多加一个对9的判断就可以了。如果是选项9，就有相应的打印操作(空格显示时间、游戏结束显示时间等等)。这样我们的子题目8和子题目9就完成了。其他的选项相对于9来说都是简单的，只要在9的基础上删去一些功能就可以了。

2.2.实验项目的文件组成

2.2.1 mine_sweeper.h

整个扫雷游戏程序的头文件，有着所有 include 的文件和所有函数的声明，本扫雷游戏程序一共定义了五十多个函数，都按照相应的功能划分部分放在了这个头文件里面。

2.2.2 mine_sweeper_menu.cpp

本文件存放了扫雷游戏的两个菜单函数，即主菜单和副菜单，主菜单用来选择游戏的选项，副菜单用来选择游戏的难度。主菜单获得 choice,副菜单获得 difficulty，这两个参数是划分该扫雷游戏程序的重要基准参数，因此整个程序由此处产生的结果为依据进行分支。

2.2.3 mine_sweeper_main.cpp

本文件存放了扫雷游戏的主函数，主函数主要有两个功能，第一是调整控制台的大小和字体，第二是调用两个 menu 函数产生分支依据，调用相应的内部数组函数和伪图形界面函数。

2.2.4 mine_sweeper_base.cpp

本文件存放了扫雷游戏程序中实现内部数组部分(1-4)的功能函数，比如有该部分“主函数”、输出函数、输入函数、打印初始雷区函数、循环游戏函数和打印输入说明函数等等，这一整个 cpp 文件和 mine_sweeper_tools.cpp 里的函数共同完成选项 1-4 中内部数组的功能实现。

2.2.5 mine_sweeper_graph.cpp

本文件存放了扫雷游戏程序中实现伪图形化界面部分(5-9)的功能函数，比如有该部分“主函数”、输出打印函数、有关鼠标的操作函数、打印初始雷区函数、循环游戏函数、打印初始的框架函数和打印输入的说明函数等等，这一整个 cpp 文件和 mine_sweeper_tools.cpp 里的函数共同完成选项 5-9 中的伪图形化界面的功能实现。

(6)mine_sweeper_tools.cpp

本文件存放了扫雷游戏程序中实现内部数组部分(1-4)和伪图形化界面部分(5-9)的共用的函数，比如初始内部数组函数、判断输赢函数和扩散雷区图函数。是两个部分都需要使用到的函数集合。

3.主要功能的实现

3.1.主要功能的函数实现

3.1.1 菜单部分：

(1) main_menu函数

该函数的作用是打印主菜单（游戏选项）并且返回一个选择(choice)

(2) sub_menu函数

该函数的作用是打印副菜单（难度选择）并且返回一个选择(difficulty)

3.1.2 内部数组部分(x表示相应的难度选择)：

(1)Inner_difficulty_x函数

该函数是mine_sweeper_base.cpp中的“主函数”，完成内部数组部分(选项1-4)功能的主要函数，输入参数为选项choice，调用内部数组中的其它函数，完成主要功能。

(2)Print_specification函数

该函数的作用是打印4选项中的各项特殊输入说明，由于多次调用，统一写为一个函数。

(3)Print_InitArray_x函数

该函数的作用是打印3.4选项游戏刚开始的雷盘，一开始的雷盘都是“X”。

(4)Color_of_Inner函数

该函数的作用是打印打开后每个点对应的雷数和该数字的颜色，输入参数为某点位置的字符，由于雷数不同的数字颜色也是不一样的，所以需要这一个函数来打印不同的颜色。

(5)Input_x函数

该函数的作用是完成内部数组部分输入功能的主要函数。输入每一次的选择位置时相应的mark数组和flag函数会发生变化，所以会在这个函数中做出相应的处理。

(6)Inner_Output_x函数

该函数的作用是完成内部数组部分输出功能的主要函数，输出的时候需要注意输出位置是否已被标记、是雷还是非雷，所以需要做出相应的判断。

(7)Inner_game_x函数

该函数的作用是完成内部数组部分3.4选项循环扫雷游戏功能的主要函数。

3.1.3 伪图形界面部分(x表示相应的难度选择)：

(1)Pseudographics_dfficulty_x函数

该函数是 mine_sweeper_graph.cpp 中的“主函数”，完成伪图形化界面部分(选项 5-9)功能的主要函数，输入参数为选项 choice，调用伪图形化界面部分中的其它函数，完成主要功能。

(2)Print_framework_x函数

该函数的作用是打印伪图形化界面部分的框架，利用循环操作，打印不同大小的框架。

(3)Print_InitGraph_x函数

该函数的作用是打印伪图形化界面部分的初始雷区显示，一开始的雷区都是橙色的。

(4)Color_of_Graph函数

该函数的作用是打印打开后每个点对应的雷数和该数字的颜色，输入参数为某点位置的字符，由于雷数不同的数字颜色也是不一样的，所以需要这一个函数来打印不同的颜色，其次还需要打印它的背景色(灰色)，这一点与内部数组部分不同。

(5)Print_Leftboom函数

该函数的作用是打印剩余雷数，要求在对应的位置上打印剩余雷数。

(6)Print_markboom_map_x函数

该函数的作用是打印被标记的雷图，因为被标记后打印出时不一样的颜色(红色)，所以这里写一个函数来完成该选项功能。

(7)Graph_Output_x函数

该函数的作用完成伪图形部分的打印功能函数，需要考虑输出位置是否已被标记、是雷还是非雷，所以需要做出相应的判断。

(8)Mouse_Operation_x函数

该函数的作用是完成伪图形部分的鼠标功能函数，鼠标主要有三个功能：左键单击、右键单击和鼠标移动，同时引入了在子题目8和9中的键盘空格和ESC的输入。

(9)Graph_game_x函数

该函数的作用是完成伪图形部分的8.9选项循环扫雷游戏功能的主要函数。

3.1.4 共用函数部分：

(1)Initarr_x函数

该函数的作用是初始化雷和每个点周围的雷数，注意我们要求第一次输入的位置是0(自身非雷且周围八个位置无雷)，所以传入参数还要有第一次输入的坐标，然后再进行判断(即在除了选项1的部分在加入第一次输入点周围八个点不能是雷，少雷便重新生成)。

(2)Reset_markarr_x函数

该函数的作用是将mark数组重置(标记两次后为无标记)。

(3)Expand_map_x函数

该函数的作用是完成难度1“打开”操作的函数，采用递归的方法，向被“打开”的周围八个点继续使用该函数，直到遇到递归的终止条件(到达边界或者到达所有非0但不是雷的位置),我还要注意依据打开的点无需重复扩散，所以我引入了flag数组来记录这个点是否已经被扩散过。

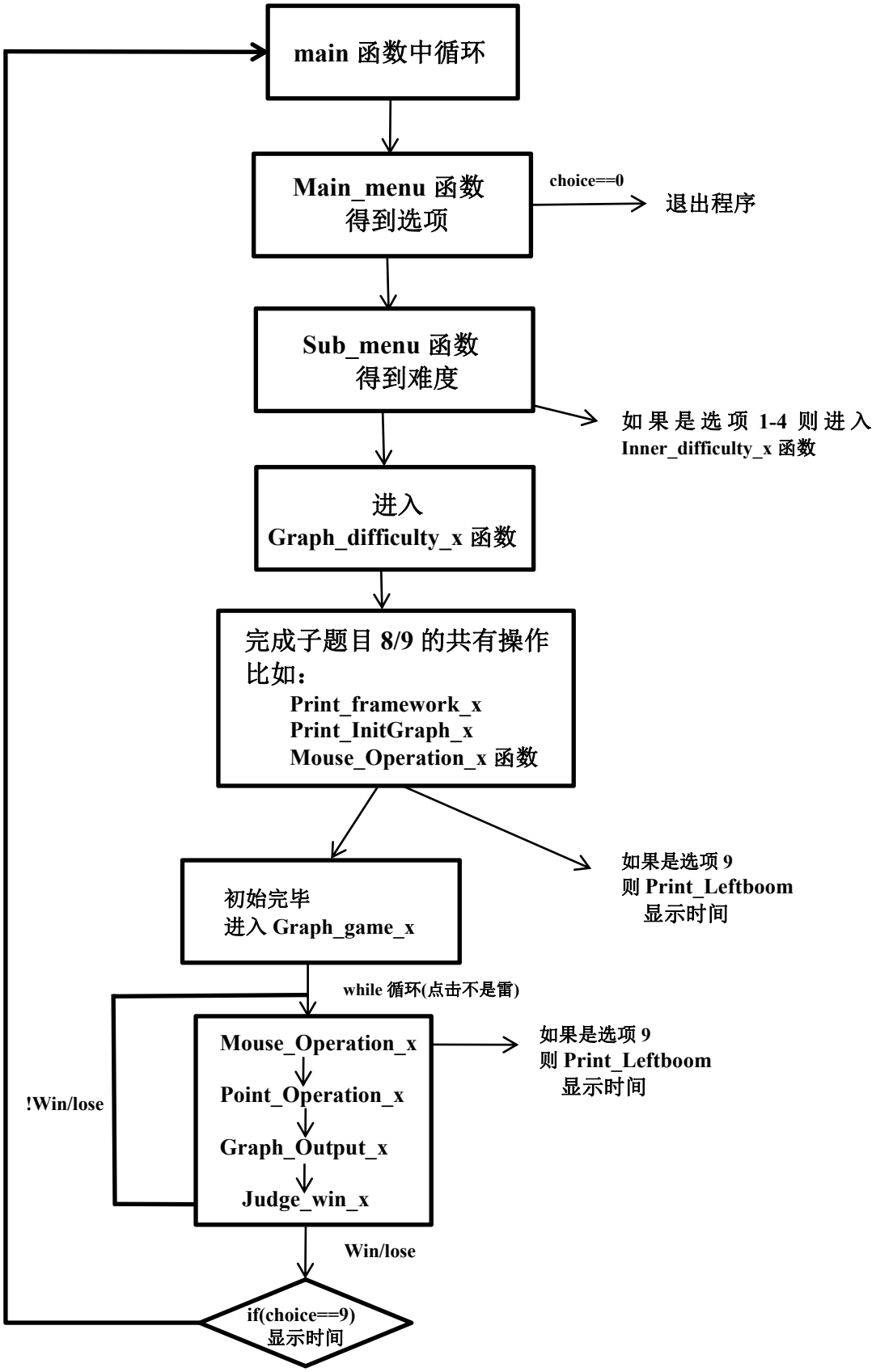
(4)Point_Operation_x函数

该函数的作用是完成对按下某点后的变化操作，分为好几种情况：是否继续打开，mark数组(标记)、flag数组(是否被扩散)和cover数组(最终的输出数组)的变化。

(5)Judge_win_x函数

该函数的作用是判断游戏是否胜利，如果所有的雷都显示出来，则表示胜利。

3.2.主要功能的流程框架(以子题目8/9为准)



4.调试中遇到的问题

4.1.问题1：有关颜色和控制台大小的问题

本次大作业涉及到许多颜色的打印和控制台大小和字体的控制。首先来说说颜色打印。在内部数组和伪图形的数字中，不同的数字有着不一样的颜色，这就比较烦人，需要不断地setcolor和找相应的颜色。(我数了数，这次要用的颜色有快二十种)。很多时候这个点颜色用对了，那个选项的颜色又莫名其妙不太对，这需要非常认真地比对老师给的cmd_consoles_tools里的颜色。最后我写了两个关于颜色的函数解决了大多的问题。还有就是，颜色变了之后一定要给它重新setcolor黑白回来！不然的话...再说说控制台大小和字体，这次作业每一个选项也有不一样窗口大小，每一种难度也有不一样的窗口大小，如果那个地方忘记把控制台大小设置回来，就比如我有一次再打印大框架时还是用小的窗口，电脑直接死机了，(www只能重来)。所以要非常细心地再每一个场景切换中调整相应的控制台大小。

4.2.问题2：有关解决扩散函数的问题

扩散函数算是此次作业中的一个核心点了，一开始我设置的扩散函数要么发生越界，要么发生重叠，在那次高程课听了老师给我们的建议之后，我将每一种难度的数组大小都扩大了一圈，这样子可以解决我在扩散中发生越界的问题，这个问题就解决了；其次，有关于我的扩散函数有些时候对有些时候不对，我在debug的过程中发现应该是它又扩散回来了，因为我是对于每一个点的八个周围点再进行扩散，哪八个周围点的周围又有刚刚扩散过的点，所以会发生一些问题。解决方法是我又设置了一个数组flag，记录某一个点是否已经被扩散过，如果被扩散过就不动。

4.3.问题3：有关打印、鼠标位置的问题

本次大作业涉及到多个打印问题。那些最基础的打印内部数组啥的已经算是简单的了。比较麻烦的是打印框架那一部分。我一开始是一半打表一半循环的，后来老师要求不能打表，我就只能重新写了。打印框架最主要的是要数清楚某一个线框到底又多少个，它又占多大的位置等等，这盯着那么小的线框数是真的挺费劲的，还好最后还是成功地用循环打印出来了(两层循环)，第一层循环打印长方向，第二层循环打印宽方向；鼠标的定位我觉得是这个程序中最烦人的，因为要通过X和Y所在的行和列(i和j)的位置来确定当前鼠标位置是否合法，这需要通过一定的数学计算，(最主要的还是要在那里数位置，有些时候数错了还得重数)，最后得到相应的数学公式，完成鼠标的合法性检验。还有一个打印位置：就是最后的那个“按回车键继续...”，也是有讲究的，这句话在最后要在哪里打印出来我们还需要通过分析每一个部分最后的光标位置，然后gotoxy到相应的位置上去再进行输出。

4.4.问题4：有关代码重用时遇到的小差错

代码的重用在本次作业的程序里得到了很大的体现。我这次写了程序按照难度系数分为了三个部分，即在每一个函数的后面加上一个后缀_1、_2或者_3表示当前函数是处理哪一种难度系数的。很多情况下我只要写完一种难度系数，剩余的两种难度系数只要复制黏贴，改一些长宽参数即可。都是有些函数是不能只复制黏贴即可，就比如内部数组部分的输入函数，由于长和框都发生变化了，所以相对应的输入行数、列数也会发生变化，而在内部数组部分中的长是分为两个部分，数字部分和小写字母部分，这一点要注意到，不然的话在输入的时候某些行列输入不进去。代码重用主打的是一个细心，否则某个参数不对劲了就会显示不出正确的结果然后debug很久。

5. 心得与体会

5.1.本次作业的经验教训和心得体会

本次高程大作业作业量很大，包括我在代码里写的几百行的注释，扫雷游戏的实现程序我一共写了两千多行；思考量也很大，我花了整整两周时间完成扫雷游戏程序及其报告。我的收获很多，主要的收获如下陈述：首先，如何划清扫雷游戏程序的每一个部分，我们要以什么来进行分块处理，我最后选择了按照难度进行总的分类，再通过两种不一样的部分(内部数组部分和伪图形界面部分)进行分cpp文件的处理。其次，就是对cmd_console_tools.cpp中给的函数的更加熟练的应用，此次高程大作业涉及到多次的颜色变化、不同的颜色变化、控制台大小的不断变换，这些都是需要用到cmd_console_tools.cpp中给的函数来一步一步地完成的(同时需要注意到所给的参数来达成不同的颜色和大小效果)，而且，这次高程大作业新增了对于鼠标函数的使用，左键、右键和移动操作的使用都是主要需要掌握的。鼠标函数的使用也是我的一大收获；然后就是体会到利用多个函数写复杂程序的巧妙之处，本次的高程大作业，我蛮打蛮算写了五十多个函数，在主函数中调用各个函数，在每一个cpp中的“主函数”中调用它们相应的解函数，这是灵活而又巧妙的。最后一点非常重要，那就是培养了我写代码时的心态和积累了debug的经验。要细心地对待每一行代码，同时在debug时要准确地定位到出错代码的大致位置，通过调试、设置断点不断积累debug经验，这使得我在后续的编写代码中有了更多的解决问题的方法。

5.2.函数的分解与使用

我考虑了每一个小题的的前后关联关系。在写扫雷游戏程序一开始的时候，我就先对整体的框架进行了构思，确保在每一个函数中都尽可能地覆盖更多的功能。就比如初始化内部数组的那一个函数，由于之后的选项都是要求一开始点开的位置不是雷并且它的周围八个位置也都不能是雷，使用在写选项1的时候，我就将这一点考虑下来了，写了一共每一个选项都通用的Init_array函数。我尽可能地做到在后面的小题中用到前面小题的代码，我之前说过了，每一个函数我都是考虑到每一个选项的功能实现的，在每一个函数内部我都加上了if-else判断来分析每一种选项可能会有的操作，所以我做到了在后面的小题中用到前面小题的代码，有些代码我就直接复制黏贴改一改相关的参数即可。

接下来时我对于如何更好地重用代码的思考：

首先还是需要我们对于这些相似的操作功能要有深入的了解，这个时候最好要画流程图，把每一个功能的具体实现过程大致显示出来，因为有一些功能的前几部是相似的，我们只需要改一改参数即可；其次，我认为在重用代码的时候要做到函数输入参数的完整化，这句话的意思是一个函数可能支持多种功能，不同的功能选项的输入大部分是相同的，可能有一些选项需要输入的参数比较多，这个时候我们就可以在函数输入参数的时候将这些多的参数也要一起输入进去，在函数里面再进行判断，没有用到对参数用if-else语句来判断即可。最后，在分析问题的时候要注意分层分块，将一个比较巨大的问题用多个函数有序地用逻辑顺序表现出来，不要受到无关参数的影响，很多看似需要用多个函数表达的问题可能这多个函数之间存在相似的逻辑，要善于将这些类似的操作归类成同一函数。最后，遇到一个像本次扫雷游戏实现这样复杂的程序的时候，我们最好找到划分区块的依据(本道大作业的划分依据是难度系数和两种不同的实现方式)，然后一步步进行推理分析，在写base部分时就不要管graph部分了，写到graph部分自然就发现base部分的有些函数是可以重用的，这样就好起来了。

6. 附件：源程序

(主要是子题目8和9的实现，所以没有mine_sweeper_base.cpp的部分，且以难度1为示例)

mine_sweeper_main.cpp

```
int main()
{
	cct_setfontsize("新宋体", 24);
	cct_setcolor(0, 7);
	cct_setconsoleborder(100, 30, 100, 30);
	while (1)
	{
		int choice = Main_menu();
		if (choice == 0)
		{
			putchar('0');
			break;
		}
		if (choice == 1 || choice == 2 || choice == 3 ||
choice == 4)
		{
			cct_cls();
			int difficulty = Sub_menu();
			cct_cls();
			if (difficulty == 1)
				Inner_difficulty_1(choice);
			else if (difficulty == 2)
				Inner_difficulty_2(choice);
			else
				Inner_difficulty_3(choice);
		}
		if (choice == 5 || choice == 6 || choice == 7 ||
choice == 8 || choice == 9)
		{
			cct_cls();
			int difficulty = Sub_menu();
			cct_cls();
			if (difficulty == 1)
			    Pseudographics_dfficulty_1(choice);
			else if (difficulty == 2)
			    Pseudographics_dfficulty_2(choice);
			else
			    Pseudographics_dfficulty_3(choice);
		}
		cct_setcolor(0, 7);
		if (choice == 5 || choice == 6)
			cct_gotoxy(0, 52);
		cout << "按回车键继续...";
		while (_getch() != '\r');
		cct_cls();
		cct_setfontsize("新宋体", 24);
		cct_setconsoleborder(100, 30, 100, 30);
	}
	cout << endl;
	return 0;
}
```

mine_sweeper_menu.cpp

```
int Main_menu()
{
	cout << "----------------------------------------" <<
endl;
	cout << "1.选择难度并显示内部数组" << endl;
	cout << "2.输入初始位置并显示被打开的初始区
域" << endl;
	cout << "3.内部数组基础版" << endl;
	cout << "4.内部数组完整版（标记、运行时间）" <<
endl;
	cout << "5.画出伪图形化框架并显示内部数据"
<< endl;
	cout << "6.检测鼠标位置和合法性（左键单击退
出）" << endl;
	cout << "7.鼠标选择初始位置并显示被打开的初
始区域" << endl;
	cout << "8.伪图形界面基础版" << endl;
	cout << "9.伪图形界面完整版" << endl;
	cout << "0.退出游戏" << endl;
	cout << "----------------------------------------" <<
endl;
	cout << "[请选择] :";
	cct_gotoxy(11, 12);
	while (1)
	{
		char keydown = _getch();
		if (keydown >= '0' && keydown <= '9')
			return (keydown - '0');
		else
			continue;
	}
}
int Sub_menu()
{
	cout << "请选择难度：" << endl;
	cout << "   1.初级( 9*9   - 10 颗雷)" << endl;
	cout << "   2.中级(16*16 - 40 颗雷)" << endl;
	cout << "   3.高级(16*30 - 99 颗雷)" << endl;
	cout << "请输入[1..3]：";
	cct_gotoxy(14, 4);
	while (1)
	{
		char keydown = _getch();
		if (keydown >= '1' && keydown <= '3')
		{
			cct_cls();
			return (keydown - '0');
		}
		else
			continue;
	}
}
```

mine_sweeper_tools.cpp

```
void Reset_markarr_1(char mark_1[11][11])
{
	for (int i = 1; i < 10; i++)
		for (int j = 1; j < 10; j++)
			if (mark_1[i][j] == 2)
				mark_1[i][j] = 0;
}
void Initarr_1(char boomarr_1[11][11], int keydown1, int
keydown2, int choice)
{
	srand((unsigned int)(time(0)));
```

```
	for (int boom_num = 0; boom_num < Boom_num_1;)
	{
		boom_num = 0;
		boomarr_1[rand() % 9 + 1][rand() % 9 + 1] = '*';
		if (choice == 2 || choice == 3 || choice == 4 || choice
== 7 || choice == 8 || choice == 9)
		{
			for (int crosswise = keydown1 - 1; crosswise
<= keydown1 + 1; crosswise++)
				for (int vertical = keydown2 - 1; vertical
<= keydown2 + 1; vertical++)
					if (boomarr_1[crosswise][vertical]
== '*')

	boomarr_1[crosswise][vertical] = 0;
		}
		for (int i = 1; i < 10; i++)
			for (int j = 1; j < 10; j++)
				if (boomarr_1[i][j] == '*')
					boom_num++;
	}
	for (int i = 1; i < 10; i++)
	{
		for (int j = 1; j < 10; j++)
		{
			int count_boom = 0;
			if (boomarr_1[i][j] != '*')
			{
				for (int crosswise = i - 1; crosswise <= i
+ 1; crosswise++)
					for (int vertical = j - 1; vertical <=
j + 1; vertical++)
						if
(boomarr_1[crosswise][vertical] == '*')
							count_boom++;
				boomarr_1[i][j] = count_boom + '0';
			}
		}
	}
}
bool Judge_win_1(char boomarr_1[11][11], char
cover_1[11][11])
{
	for (int i = 1; i < 10; i++)
		for (int j = 1; j < 10; j++)
			if (cover_1[i][j] == 'X')
				if (boomarr_1[i][j] != '*')
					return false;
	return true;
}
void Expand_map_1(char boomarr_1[11][11], char
cover_1[11][11], char flag_1[11][11], char mark_1[11][11], int
keydown1, int keydown2)
{
	for (int i = keydown1 - 1; i <= keydown1 + 1; i++)
	{
		for (int j = keydown2 - 1; j <= keydown2 + 1; j++)
		{
			if (flag_1[i][j] == 1)
			{
				if (mark_1[i][j] == 1)
					cover_1[i][j] = 'X';
				else
					cover_1[i][j] = boomarr_1[i][j];
				continue;
			}
			if (flag_1[i][j] == 0)
			{
				if (mark_1[i][j] == 1)
					cover_1[i][j] = 'X';
				else
					cover_1[i][j] = boomarr_1[i][j];
				flag_1[i][j] = 1;
			}
			if (boomarr_1[i][j] == '0')
				Expand_map_1(boomarr_1, cover_1,
flag_1, mark_1, i, j);
		}
	}
	for (int i = 1; i < 10; i++)
		for (int j = 1; j < 10; j++)
			if (cover_1[i][j] == 0)
				cover_1[i][j] = 'X';
}
void Point_Operation_1(char boomarr_1[11][11], char
cover_1[11][11], char flag_1[11][11], char mark_1[11][11],
int& keydown1, int& keydown2)
{
	if (boomarr_1[keydown1][keydown2] == '0' &&
mark_1[keydown1][keydown2] == 0)
	{
		Reset_markarr_1(mark_1);
		Expand_map_1(boomarr_1, cover_1, flag_1,
mark_1, keydown1, keydown2);
	}
	else
	{
		if (mark_1[keydown1][keydown2] == 1)
		{
			cover_1[keydown1][keydown2] = 'X';
			flag_1[keydown1][keydown2] = 1;
		}
		else if (mark_1[keydown1][keydown2] == 2)
			cover_1[keydown1][keydown2] =
cover_1[keydown1][keydown2];
		else
		{
			cover_1[keydown1][keydown2] =
boomarr_1[keydown1][keydown2];
			flag_1[keydown1][keydown2] = 1;
		}
	}
}
```

mine_sweeper_graph.cpp

```
void Color_of_Graph(char ch)
{
	switch (ch)
	{
	case '0':
		cct_setcolor(7, 7);
		break;
	case '1':
		cct_setcolor(7, 1);
		break;
	case '2':
		cct_setcolor(7, 2);
		break;
	case '3':
```

```
            cct_setcolor(7, 3);
            break;
        case '4':
            cct_setcolor(7, 4);
            break;
        case '5':
            cct_setcolor(7, 5);
            break;
        case '6':
            cct_setcolor(7, 6);
            break;
        case '7':
            cct_setcolor(7, 7);
            break;
        case '8':
            cct_setcolor(7, 8);
            break;
        case 'X':
            cct_setcolor(6, 6);
            break;
        case '*':
            cct_setcolor(7, 0);
            break;
        }
        if (ch == '0' || ch == 'X')
            cout << ' ';
        else   cout << ch;
        cct_setcolor(0, 7);
}
void Print_framework_1()
{
        cout << endl;
        for (int i = 0; i < 9; i++)
            cout << setw(6) << i;
        cout << endl << "    ";
        cct_setcolor(15, 0);      cout << "╔";
        for (int i = 0; i < 8; i++)
            cout << "═══╦";
        cout << "═══╗" << endl;
        cct_setcolor(0, 7);       cout << "    ";
        cct_setcolor(15, 0);      cout << "║";
        for (int i = 0; i < 9; i++)
            cout << "       ║";
        cout << endl;
        cct_setcolor(0, 7);       cout << "    ";
        cct_setcolor(15, 0);      cout << "║";
        for (int i = 0; i < 9; i++)
            cout << "       ║";
        cout << endl;
        for (int i = 0; i < 8; i++)
        {
            cct_setcolor(0, 7);       cout << "    ";
            cct_setcolor(15, 0);      cout << "╠";
            for (int j = 0; j < 8; j++)
                cout << "═══╬";
            cout << "═══╣" << endl;
            cct_setcolor(0, 7);       cout << "    ";
            cct_setcolor(15, 0);      cout << "║";
            for (int j = 0; j < 9; j++)
                cout << "       ║";
            cout << endl;
            cct_setcolor(0, 7);       cout << "    ";
            cct_setcolor(15, 0);      cout << "║";
            for (int i = 0; i < 9; i++)
                cout << "       ║";
            cout << endl;
        }
        cct_setcolor(0, 7);       cout << "    ";
        cct_setcolor(15, 0);      cout << "╚";
        for (int i = 0; i < 8; i++)
            cout << "═══╩";
        cout << "═══╝" << endl;
        cct_setcolor(0, 7);
        for (int i = 0; i < 9; i++)
        {
            char ch = 'A';
            cct_gotoxy(0, 3 * (i + 1) + 1);
            putchar(ch + i);
        }
}
void Print_InitGraph_1()
{
        for (int i = 1; i < 10; i++)
        {
            for (int j = 1; j < 10; j++)
            {
                cct_gotoxy(6 * j - 2, 3 * i);
                cct_setcolor(6, 6);
                cout << "       ";
                cct_gotoxy(6 * j - 2, 3 * i + 1);
                cout << "       ";
                cct_setcolor(0, 7);
            }
        }
}
void Print_Leftboom(int Left_boom,int choice)
{
        if (choice == 9)
        {
            cct_gotoxy(35, 0);
            cct_setcolor(0, 14);
            if (Left_boom > 0)
                cout << setw(2) << Left_boom << " ";
            else          cout << setw(2) << 0 << " ";
            cct_setcolor(0, 7);
        }
}
void Print_markboom_map_1(char mark_1[11][11])
{
        for (int i = 1; i < 10; i++)
        {
            for (int j = 1; j < 10; j++)
            {
                if (mark_1[i][j] == 1)
                {
                    cct_gotoxy(6 * j - 2, 3 * i);
                    cct_setcolor(12, 4);      cout << "       ";
                    cct_gotoxy(6 * j - 2, 3 * i + 1);
                    cout << " ";
                    cct_setcolor(12, 7);      cout << '#';
                    cct_setcolor(12, 4);      cout << "    ";
                    cct_setcolor(0, 7);
                }
                else
                {
                    cct_gotoxy(6 * j - 2, 3 * i);
                    cct_setcolor(6, 6);
                    cout << "       ";
```

```
                        cct_gotoxy(6 * j - 2, 3 * i + 1);
                        cout << "       ";
                        cct_setcolor(0, 7);
                    }
                }
            }
}
void Pseudographics_dfficulty_1(int choice)
{
    clock_t Startpoint, Endpoint;
    Startpoint = Endpoint = clock();
    cct_setfontsize("点阵字体", 16, 8);
    cct_setconsoleborder(59, 35, 59, 35);
    char boomarr_1[11][11] = { 0 };
    char cover_1[11][11] = { 0 };
    char flag_1[11][11] = { 0 };
    char mark_1[11][11] = { 0 };
    int keydown1 = 0;
    int keydown2 = 0;
    int Keydown_ret = 0;
    int Left_boom = Boom_num_1;
    Print_framework_1();
    if (choice == 7 || choice == 8 || choice == 9)
    {
        if (choice == 8)
        {
            cct_gotoxy(0, 0);
            cout << "按Esc退出";
        }
        if (choice == 9)
        {
            cct_gotoxy(0, 0);
            cout << "按Esc退出，空格显示时间    ";
            cct_setcolor(0, 14);
            cout << "剩余雷数：" << setw(2) <<
Left_boom;
        }
        Print_InitGraph_1();
        Mouse_Operation_1(choice, keydown1, keydown2,
flag_1, mark_1, Keydown_ret, Left_boom);
        Print_Leftboom(Left_boom, choice);
        while (Keydown_ret == 2)
        {
            cct_gotoxy(39, 0);
            Endpoint = clock();
            cct_setcolor(0, 14);
            cout << "当前时间:" << (double)(Endpoint -
Startpoint) / CLOCKS_PER_SEC << "秒";
            cct_setcolor(0, 7);
            Keydown_ret = 0;
            Mouse_Operation_1(choice, keydown1,
keydown2, flag_1, mark_1, Keydown_ret, Left_boom);
            Print_Leftboom(Left_boom, choice);
        }
        if (Keydown_ret == 1)
        {
            cct_gotoxy(0, 31);
            return;
        }
        cct_gotoxy(39, 0);
        cout << "                                ";
        cct_setcolor(0, 7);
        while (mark_1[keydown1][keydown2] != 0)
        {
            Print_markboom_map_1(mark_1);
            Mouse_Operation_1(choice, keydown1,
keydown2, flag_1, mark_1, Keydown_ret, Left_boom);
            Print_Leftboom(Left_boom, choice);
            while (Keydown_ret == 2)
            {
                cct_gotoxy(39, 0);
                Endpoint = clock();
                cct_setcolor(0, 14);
                cout << "当前时间:" << (double)
(Endpoint - Startpoint) / CLOCKS_PER_SEC << "秒";
                cct_setcolor(0, 7);
                Keydown_ret = 0;
                Mouse_Operation_1(choice, keydown1,
keydown2, flag_1, mark_1, Keydown_ret, Left_boom);
                Print_Leftboom(Left_boom, choice);
            }
            if (Keydown_ret == 1)
            {
                cct_gotoxy(0, 31);
                return;
            }
            cct_gotoxy(39, 0);
            cout << "                                ";
        }
    }
    Reset_markarr_1(mark_1);
    Initarr_1(boomarr_1, keydown1, keydown2, choice);
    if (choice == 5)
        Graph_Output_1(choice, boomarr_1, cover_1,
mark_1);
    if (choice == 6)
    {
        Graph_Output_1(choice, boomarr_1, cover_1,
mark_1);
        Mouse_Operation_1(choice, keydown1, keydown2,
flag_1, mark_1, Keydown_ret, Left_boom);
    }
    if (choice == 7)
    {
        Expand_map_1(boomarr_1, cover_1, flag_1,
mark_1, keydown1, keydown2);
        Graph_Output_1(choice, boomarr_1, cover_1,
mark_1);
        cct_gotoxy(0, 32);
    }
    if (choice == 8 || choice == 9)
    {
        Expand_map_1(boomarr_1, cover_1, flag_1,
mark_1, keydown1, keydown2);
        Graph_Output_1(choice, boomarr_1, cover_1,
mark_1);
        Graph_game_1(choice, boomarr_1, cover_1, flag_1,
mark_1, keydown1, keydown2, Startpoint, Endpoint,
Left_boom);
    }
}
void Graph_Output_1(int choice, char boomarr_1[11][11], char
cover_1[11][11], char mark_1[11][11])
{
    if (choice == 5 || choice == 6)
    {
        for (int i = 1; i < 10; i++)
```

```
            {
                for (int j = 1; j < 10; j++)
                {
                    cct_gotoxy(6 * j - 2, 3 * i);
                    cct_setcolor(7, 7);        cout << "        ";
                    cct_gotoxy(6 * j - 2, 3 * i + 1);
                    cout << " ";
                    Color_of_Graph(boomarr_1[i][j]);
                    cct_setcolor(7, 7);        cout << "    ";
                }
            }
            cct_gotoxy(0, 31);
        }
        if (choice == 7 || choice == 8 || choice == 9)
        {
            for (int i = 1; i < 10; i++)
            {
                for (int j = 1; j < 10; j++)
                {
                    if (cover_1[i][j] == 'X' &&
mark_1[i][j] != 1)
                    {
                        cct_gotoxy(6 * j - 2, 3 * i);
                        cct_setcolor(6, 6);
                        cout << "        ";
                        cct_gotoxy(6 * j - 2, 3 * i + 1);
                        cout << "        ";
                        cct_setcolor(0, 7);
                    }
                    else if (cover_1[i][j] == 'X' &&
mark_1[i][j] == 1)
                    {
                        cct_gotoxy(6 * j - 2, 3 * i);
                        cct_setcolor(12, 4);
                        cout << "        ";
                        cct_gotoxy(6 * j - 2, 3 * i + 1);
                        cout << " ";
                        cct_setcolor(12, 7);      cout << '#';
                        cct_setcolor(12, 4);
                        cout << "    ";
                    }
                    else
                    {
                        cct_gotoxy(6 * j - 2, 3 * i);
                        cct_setcolor(7, 7);
                        cout << "        ";
                        cct_gotoxy(6 * j - 2, 3 * i + 1);
                        cout << " ";
                        Color_of_Graph(cover_1[i][j]);
                        cct_setcolor(7, 7);    cout << "    ";
                    }
                }
            }
        }
        cct_setcolor(0,7);
}
void Mouse_Operation_1(int choice, int& keydown1, int&
keydown2, char flag_1[11][11], char mark_1[11][11], int&
Keydown_ret, int& Left_boom)
{
        bool loop = 1;
        int X = 0, Y = 0;
        int Event, Maction;
        int keycode1, keycode2;
        cct_enable_mouse();
        cct_setcursor(CURSOR_INVISIBLE);
        while (loop)
        {
            Event = cct_read_keyboard_and_mouse(X, Y,
Maction, keycode1, keycode2);
            if (Event == CCT_MOUSE_EVENT)
            {
                cct_gotoxy(0, 30);
                bool Legical = false;
                cout << "[当前光标]";
                switch (Maction)
                {
                case MOUSE_ONLY_MOVED:
                    for (int i = 1; i < 10; i++)
                    {
                        for (int j = 1; j < 10; j++)
                        {
                            if (X >= 6 * j - 2 && X <= 6
* j + 1 && Y >= 3 * i && Y <= 3 * i + 1)
                            {
                                Legical = true;
                                char ch = 'A';
                                cout << " " <<
(char)(ch + i - 1) << "行" << j - 1 << "列  ";
                                break;
                            }
                        }
                    }
                    if (!Legical)
                        cout << "  位置非法";
                    break;
                case MOUSE_LEFT_BUTTON_CLICK:
                    for (int i = 1; i < 10; i++)
                    {
                        for (int j = 1; j < 10; j++)
                        {
                            if (X >= 6 * j - 2 && X <= 6
* j + 1 && Y >= 3 * i && Y <= 3 * i + 1)
                            {
                              if (choice == 6)
                              {
                                  loop = false;
                                  cct_gotoxy(0, 31);
                                      break;
                              }
                                  keydown1 = i;
                                  keydown2 = j;
                                  if
(mark_1[keydown1][keydown2] != 1)          Legical = true;
                                  if
(mark_1[keydown1][keydown2] == 2)
                        mark_1[keydown1][keydown2] = 0;
                            }
                        }
                    }
                    if (Legical)
                        loop = 0;
                    break;
                case MOUSE_RIGHT_BUTTON_CLICK:
                    if (choice == 8 || choice == 9)
                    {
                        for (int i = 1; i < 10; i++)
                        {
```

```
                                for (int j = 1; j < 10; j++)
                                {
                                        if (X >= 6 * j - 2 &&
X <= 6 * j + 1 && Y >= 3 * i && Y <= 3 * i + 1)
                                        {
                                                Legical = true;
                                                keydown1 = i;
                                                keydown2 = j;
                        if (flag_1[i][j] == 0 && mark_1[i][j] != 1)
                                                {

        mark_1[i][j] = 1;
        Left_boom--;
                                                }
                                        else if (mark_1[i][j] == 1)
                                                {

        mark_1[i][j] = 2;
        Left_boom++;
                                                }
                                        }
                                }
                        }
                        if (Legical)
                                loop = 0;
                        break;
                default:
                        break;
                }
        }
        else if ((choice == 8 || choice == 9) && Event ==
CCT_KEYBOARD_EVENT)
        {
                if (keycode1 == 27)
                {
                        loop = 0;
                        Keydown_ret = 1;
                        cct_gotoxy(0, 31);
                        break;
                }
                if (keycode1 == 32 && choice == 9)
                {
                        loop = 0;
                        Keydown_ret = 2;
                        break;
                }
        }
    }
    cct_disable_mouse();
    cct_setcursor(CURSOR_VISIBLE_NORMAL);
}
void Graph_game_1(int choice, char boomarr_1[11][11], char
cover_1[11][11], char flag_1[11][11], char mark_1[11][11], int
keydown1, int keydown2, clock_t Startpoint, clock_t Endpoint,
int& Left_boom)
{
    int Keydown_ret = 0;
    while (boomarr_1[keydown1][keydown2] != '*' ||
mark_1[keydown1][keydown2] != 0)
    {
        Mouse_Operation_1(choice, keydown1, keydown2,
flag_1, mark_1, Keydown_ret, Left_boom);
        Print_Leftboom(Left_boom, choice);
        while (Keydown_ret == 2)
        {
                cct_gotoxy(39, 0);
                Endpoint = clock();
                cct_setcolor(0, 14);
                cout << "当前时间:" << (double)(Endpoint -
Startpoint) / CLOCKS_PER_SEC << "秒";
                cct_setcolor(0, 7);     Keydown_ret = 0;
                Mouse_Operation_1(choice, keydown1,
keydown2, flag_1, mark_1, Keydown_ret, Left_boom);
                Print_Leftboom(Left_boom,choice);
        }
        if (Keydown_ret == 1)
        {
                cct_gotoxy(0, 31);
                return;
        }
        cct_gotoxy(39, 0);
        cout << "                                ";
        Point_Operation_1(boomarr_1, cover_1, flag_1,
mark_1, keydown1, keydown2);
        Graph_Output_1(choice, boomarr_1, cover_1,
mark_1);
        bool Judge = Judge_win_1(boomarr_1, cover_1);
        if (Judge)
        {
                cct_setcolor(0, 14);
                cct_gotoxy(0, 30);
                cout << "恭喜胜利，游戏结束    ";
                cct_setcolor(0, 7);
                if (choice == 9)
                {
                        Endpoint = clock();
                        cct_setcolor(0, 14);
                        cout << "共用时:" << (double)
(Endpoint - Startpoint) / CLOCKS_PER_SEC << "秒";
                        cct_gotoxy(35, 0);
                        cout << setw(2) << 0 << " ";
                        cct_setcolor(0, 7);
                        cct_gotoxy(0, 31);
                }
                break;
        }
        if (cover_1[keydown1][keydown2] == '*')
        {
                cct_gotoxy(0, 30);
                cct_setcolor(0, 14);
                cout << "你输了，游戏结束    ";
                cct_setcolor(0, 7);
                if (choice == 9)
                {
                        cct_setcolor(0, 14);
                        Endpoint = clock();
                        cout << "共用时:" << (double)
(Endpoint - Startpoint) / CLOCKS_PER_SEC << "秒";
                        cct_setcolor(0, 7);
                        cct_gotoxy(0, 31);
                }
        }
    }
}
```