《高级语言程序设计》实验大作业反思报告

大作业题目	扫雷的简单实现			类型	游戏
班　号	22L0201		学　号	2022113573	
所在院系	工科试验班(计算机与电子通信)	学　期	2022 年 秋季学期	任课教师	苏小红
实验类型	综合设计型				

实验目的：

- 掌握程序设计的基本算法和简单数据结构基础，能够综合运用基本控制语句、算法和数据结构，以及自顶向下、逐步求精的模块化设计方法，能够设计具有一定规模的系统级C语言程序，提高系统编程能力；
- 针对计算相关的复杂工程问题，能够使用恰当的算法和数据结构，完成计算、统计、排序、检索、匹配等相关的软件系统的构造、测试与实现；
- 掌握常用的程序调试和测试方法。

实验要求：

- 采用自顶向下、逐步求精的模块化设计思想设计一个小型信息库管理系统，或者闯关式游戏程序。
- 要求解释说明采用了什么数据结构和算法，为什么选择这种数据结构或算法，系统实现过程中遇到了哪些问题，这些问题是如何解决的，还有什么问题尚未解决，今后打算从哪几个方面进行改进，本设计的亮点和难点在哪里，实验结果如何，有哪些收获和学习体会；
- 编写程序完成以下实验大作业内容并完成实验大作业反思报告。

实验内容：

设计一个简单的扫雷程序，包括：

（1） 所有与用户交互的界面均为图形界面，由鼠标与键盘交互；

（2） 有欢迎菜单，含有“游玩”与“退出”两个按钮；

（3） 有难度选择菜单，可选“初级”,“中级”,“高级”,“自定义”四种难度；

（4） 有存档功能，可在难度选择菜单里选择加载存档；

（5） 对存档有简易的加密，防止用户直接从存档文件中获取信息；

（6） 加载存档时，对存档内容进行判断，防止存档(人为)损坏；

（7） 有游戏界面，包括功能：游戏主体，显示 emoji 图标，重试功能，返回难度选择菜单，保存，退出，帮助，高分榜功能，以及显示时间与剩余雷数；

（8） 对游戏本体，有以下交互：

鼠标左键单击未打开格子以打开格子；

鼠标左键单击数字以打开周围格子，当且仅当周围旗帜数等于该数字；

鼠标右键点击未打开格子可改变标记，无→旗帜→问号→无，循环；

当点开的格子周围没有雷时，自动打开周边格子；

当游戏结束时，忽略用户的操作；

（9） 有游戏音效，在游戏开始、胜利、失败时分别播放对应音频；

（10） 在用户未保存存档就软退出（使用退出按钮）时，提醒用户保存；

（11） 游戏胜利后，更新高分榜；

（12） 特别地，所有的按钮以及游戏的格子都有按下的动画效果，且仅当用户鼠标按键在按钮/格子范围内抬起才执行对应操作，相当于给了后悔的机会；

实验环境：

操作系统：Win11

集成开发环境：Codeblocks20.03 或 Visual Studio Code

外部库：Easy graphic engine(ege)

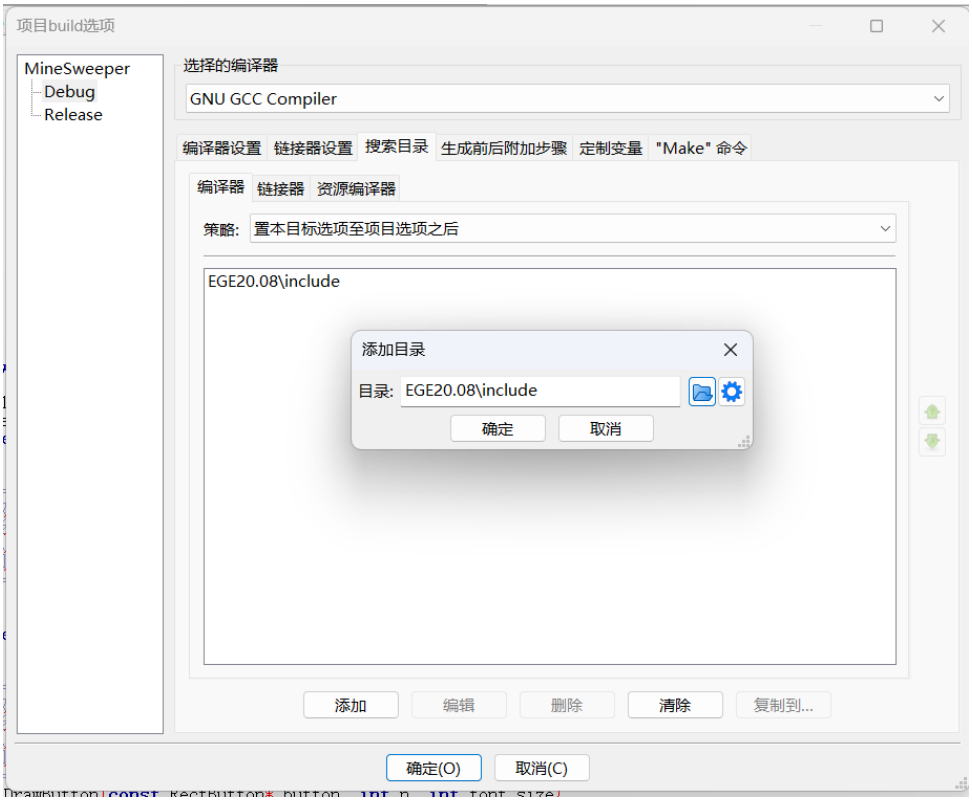
以下针对 Codeblocks20.03 环境(17.12 也可)进行 ege 的配置

1.从 ege 官网下载文件 ege20.08_all.7z，或使用本压缩包里自带的

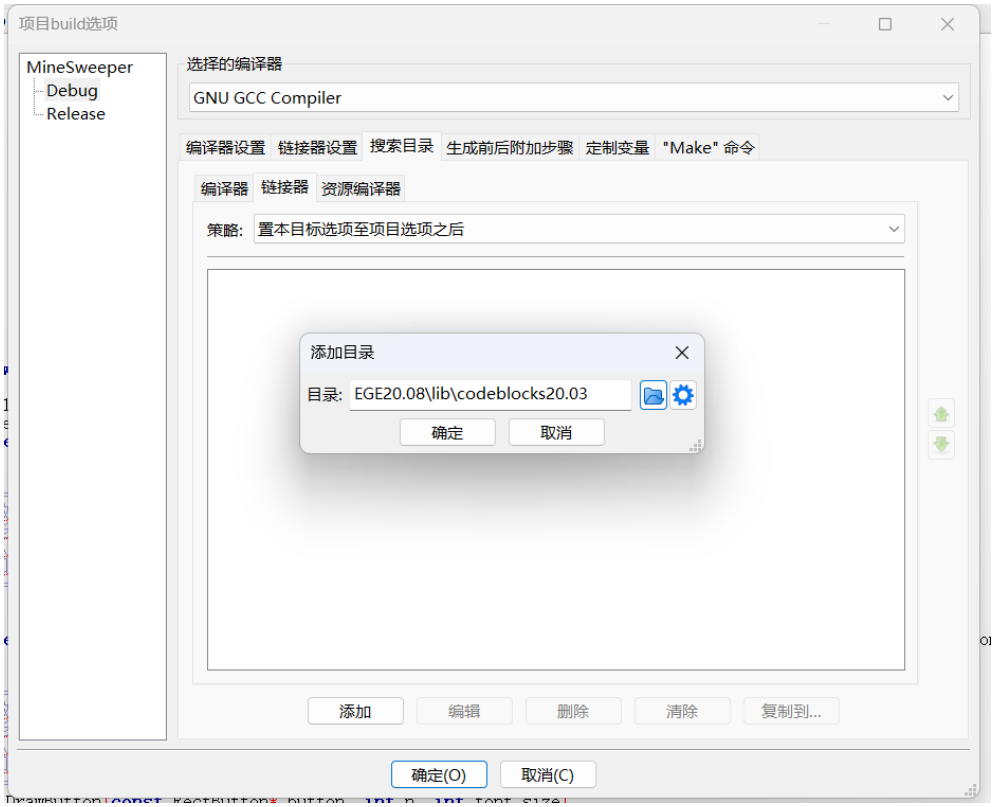
2.解压

3.启动 CodeBlocks，点击项目→构建选项。

4.在左侧栏的 Debug 与 Release 中，均打开搜索目录设置，选择编译器，添加：解压得到文件的 include 文件夹目录。



5.在左侧栏的 Debug 与 Release 中，均打开搜索目录设置，选择链接器，根据 Codeblocks 的版本选择添加：解压得到文件的 lib 文件夹中的 codeblocks20.03 文件夹或 codeblocks17.12 文件夹的目录。



6.在左侧栏的 Debug 与 Release 中，均打开链接器设置，根据 Codeblocks 的版本选择添加链接库：

17.12 版本：复制以下行，选择添加按钮，粘贴，并回车

libgraphics.a; libgdi32.a; libimm32.a; libmsimg32.a; libole32.a; liboleaut32.a; libwinmm.a; libuuid.a; libgdiplus.a

20.03 版本：复制以下行，选择添加按钮，粘贴，并回车

libgraphics64.a; libgdi32.a; libimm32.a; libmsimg32.a; libole32.a; liboleaut32.a; libwinmm.a; libuuid.a; libgdiplus.a

输入输出设计：

程序的输入有鼠标事件，自定义难度时的地图参数输入，存档文件的读取，高分榜文件的读取，以及图片文件的加载

鼠标事件为 ege 中的 mouse_msg 类型，仅对一定坐标范围内的鼠标事件相应；
对所有的鼠标事件，在用户按下鼠标时仅显示按钮/格子按下的动画，不执行操作；
仅当鼠标按键抬起时，才对鼠标抬起瞬间时鼠标所在的位置进行相应的响应操作；
这么设计，是为了让用户能有后悔的机会(即鼠标按下后，发现点错了，只要将鼠标移至他处松开即可)

地图参数的输入使用了 ege 中的 sys_edit 输入框，要求输入均为整型，其中，地图行数在 9-24 范围，列数在 9-30 范围，雷数在 10-668 范围；
当输入类型错误，或超出范围，或雷数大于等于行数乘以列数时，判定输入错误，弹窗(使用 MessageBoxA 函数实现)提醒用户，并清空输入框内容；

地图存档文件为 txt 格式，存档构成为：
前 6 行：地图行数，地图列数，雷数，旗帜数，打开的格子数，时间。均为加密后的；
第 7 行，为前六行的和，检验用；
第 8 到 7+map_row*map_col 行：为每个格子的数据，为：
由数字(0-8 为周边雷数,15 为雷)，是否按下(0-1)，是否被打开(0-1)，标记状态(0-2)
一同加密后的结果；
存档文件读入时，判断存档是否损坏：
1. 前六行之和不等于第七行；
2. 实际地图数据与前六行的简略数据不符。
当存档损坏时，弹窗提醒用户存档损坏，后留在难度选择菜单中。
当存档不存在时，弹窗提醒用户存档不存在，后留在难度选择菜单中。

高分榜文件为 txt 格式，共六行，两行为一组，有三组，分别为初级、中级、高级的高分榜数据。
每两行中，第一行为用户名，第二行为所用时间；
当高分榜文件不存在或读入错误时，用户名初始化为”-“,时间设为 99999。
不对高分榜文件进行加密处理

图片文件有 gif 格式与 png 格式，当图片读取错误时，弹窗提醒，并结束进程

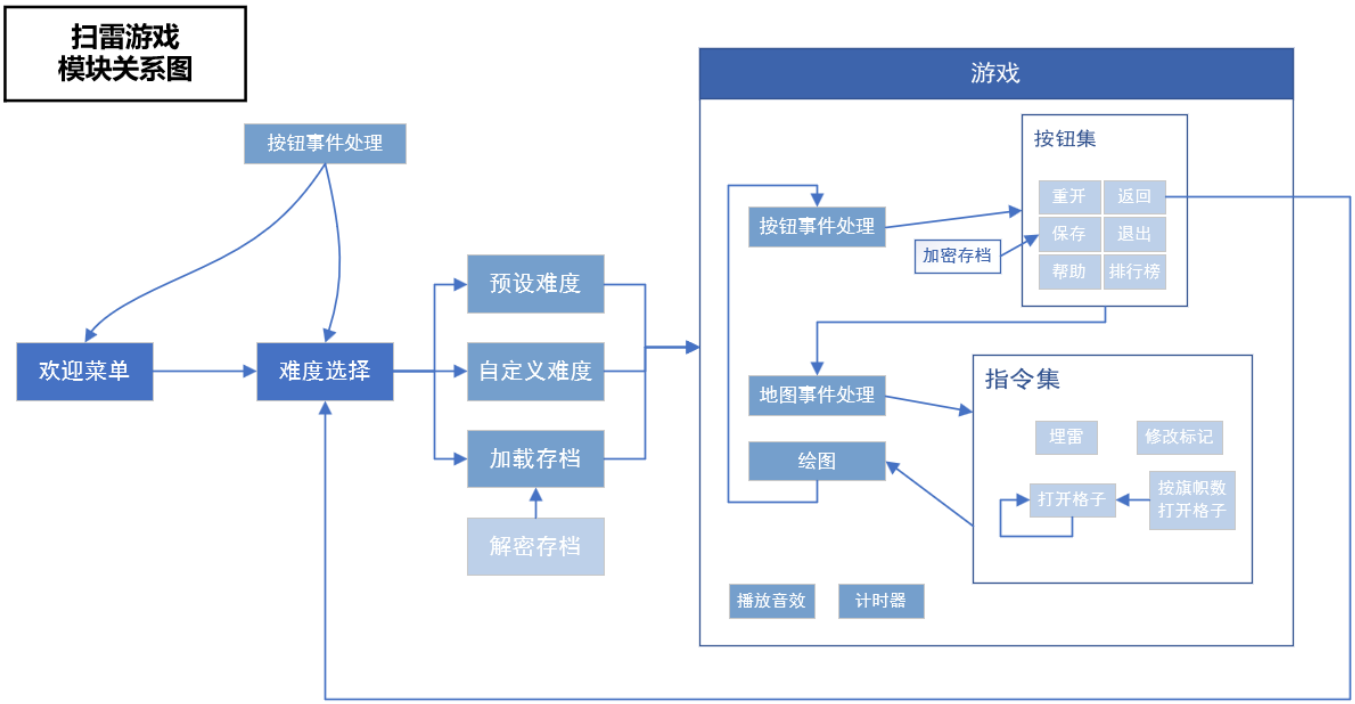
程序的输出有图形界面，存档文件的保存，高分榜文件的保存

对图形界面，按 60 帧刷新，使用缓冲区技术，防止卡顿。
对存档保存，保存失败时，弹窗提醒；
对高分榜文件保存，保存失败时，弹窗提醒；

系统设计与实现:

1. 系统功能模块划分

对系统进行自顶向下的模块分解，画出系统各个功能模块之间的结构图如下：



2. 函数功能和外部接口设计

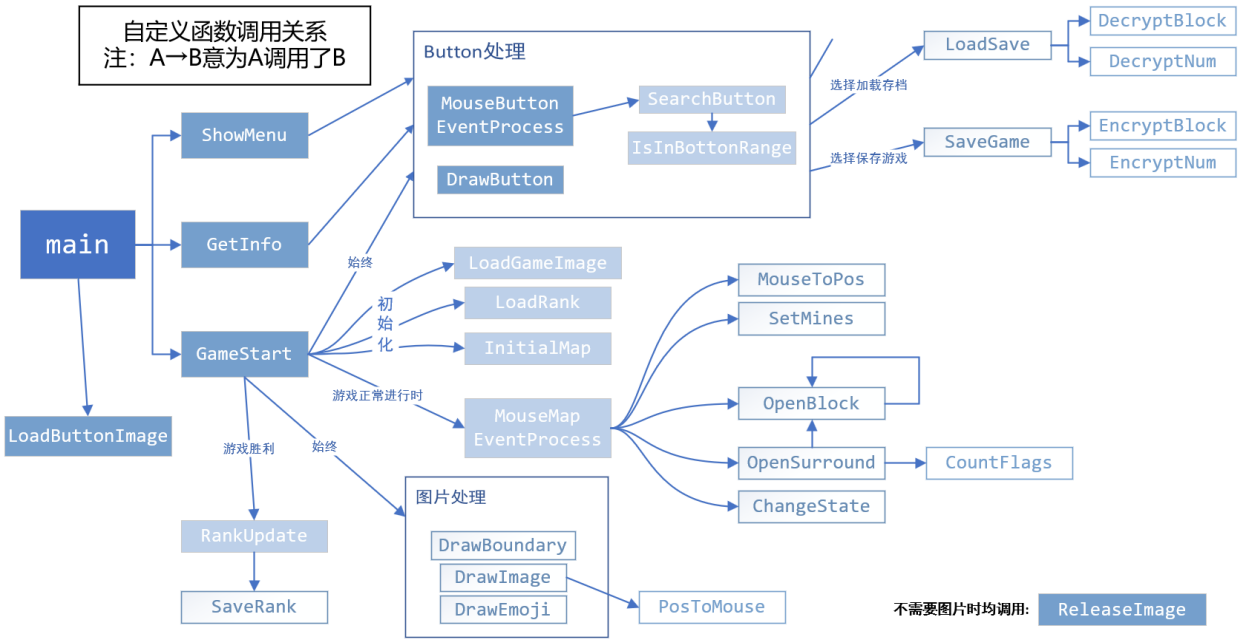
本系统总计设计了31个函数，每个函数的功能和接口设计如下表所示：

序号	函数名	函数功能	函数参数	函数返回值
1	LoadButtonImage	创建并加载按钮会使用到的图片	存放按钮图片的数组(PIMAGE*)	无
2	LoadGameImage	创建并加载游戏中会使用到的图片	存放游戏图片的数组(PIMAGE*)	无
3	ReleaseImage	释放图片数组的内存	需要释放的图片数组(PIMAGE*) 图片数量(int)	无
4	SaveGame	保存当前未完成的游戏	地图信息结构体的指针(MAP_INFO*) 地图本身(BLOCK(*)[32])	0 为保存成功 1 为保存失败
5	LoadSave	加载存档信息	地图信息结构体的指针(MAP_INFO*) 地图本身(BLOCK(*)[32])	0 为加载成功 1 为加载失败
6	RankUpdate	尝试更新排名信息	排名信息数组(RANK_INFO*) 地图信息结构体指针(MAP_INFO*) 所用时间(int)	无
7	LoadRank	加载排名信息	排名信息数组(RANK_INFO*)	无
8	SaveRank	保存排名信息	排名信息数组(RANK_INFO*)	无

9	MouseToPos	将鼠标对窗口的相对坐标转换为地图坐标	鼠标相对于窗口的坐标(x, y) (int,int)	对应的地图坐标 (MAP_POS)
10	PosToMouse	将地图坐标转换为鼠标对窗口的相对坐标(地图坐标对应格的左上角的像素坐标)	对应的地图坐标(MAP_POS) 用于接受鼠标相对于窗口坐标的指针(&x, &y) (int*,int*)	无
11	InitialMap	初始化地图，将地图清空	无	无
12	SetMines	埋雷，并更新雷周边的数字	用户点击的地图坐标 (MAP_POS) 地图信息结构体指针 (MAP_INFO*)	无
13	OpenBlock	打开格子，若为空格，递归打开周边格子	用户点击的地图坐标 (MAP_POS) 指向游戏状态的指针 (int*) 地图信息结构体指针 (MAP_INFO*)	无
14	ChangeState	改变格子标记状态，按无标记->旗帜->问号循环切换	用户点击的地图坐标对应的格子的地址(BLOCK*) 地图旗帜数地址(int*)	无
15	OpenSurround	打开周边的格子，当且仅当周边的旗帜数等于该格的数字	用户点击的地图坐标 (MAP_POS) 游戏状态指针(int*) 地图信息结构体指针 (MAP_INFO*)	无
16	CountFlags	数出周围的旗帜数	用户点击的地图坐标 (MAP_POS)	周围的旗帜数(int)
17	EncryptBlock	将地图格子的信息进行加密	地图格子(BLOCK)	加密结果(unsigned)
18	EncryptNum	将数字进行加密	数字(unsigned)	加密结果(unsigned)
19	DecryptBlock	将地图格子的信息进行解密	加密结果(unsigned)	解密结果(BLOCK)
20	DecryptNum	将数字进行解密	数字(unsigned)	解密结果(unsigned)
21	ShowMenu	显示欢迎菜单，及其按钮与对应功能	无	用户触发按钮的 id(int)
22	GetInfo	显示选择菜单，及其按钮与对应功能，主要功能是获取游戏地图的数据，以及选择加载存档	地图信息结构体指针 (MAP_INFO*)	0 为开启新游戏 1 为继续游戏
23	GameStart	包括扫雷游戏的主要内容	地图信息结构体指针 (MAP_INFO*) 游戏类型(int，0 为开启新游戏；	0 为退出 1 为返回难度设置界面

			1 为继续上局游戏，需读取文件)	
24	DrawButton	按指定字体大小绘制所有按钮	按钮数组(RectButton*) 按钮数量(int) 字体大小(int)	无
25	SearchButton	搜索输入坐标所对应的按钮 id	坐标 x，y(int,int) 按钮数组(RectButton*) 按钮数量(int)	输入坐标所对应的按钮 id(int) -1 为无对应按钮
26	IsInBottonRange	判断坐标是否在按钮范围内	需要判断的坐标(x，y 值)(int,int) 与需要判断的按钮(RectButton*)	bool 类型，表明坐标是否在按钮范围内
27	MouseButtonEventProcess	处理鼠标事件，反映在按钮上	鼠标消息(mouse_msg) 按钮数组(RectButton*) 按钮数量(int)	用户选择的按钮 id(int) -1 为未选择
28	MouseMapEventProcess	处理鼠标事件，反映在地图上	鼠标消息(mouse_msg) 雷是否已埋下(bool) 游戏状态指针(int*) 地图信息结构体指针(MAP_INFO*)	鼠标消息中鼠标位置对应的地图坐标(MAP_POS)
29	DrawBoundary	绘制网格	地图信息结构体指针(MAP_INFO*)	无
30	DrawImage	绘制网格图片	游戏图片数组(PIMAGE*) 地图信息结构体指针(int*) 游戏运行状态(int)	无
31	DrawEmoji	绘制 Emoji 图片	游戏图片数组(PIMAGE*) 是否展示惊讶的布尔值(bool) 游戏运行状态(int) 地图的列数(int)	无

各个函数之间的调用关系如下所示：



3. 数据结构

给分范围为：结构体数组+指针数组

使用二维 BLOCK 结构体数组存储地图信息；

使用 IMAGE 类指针(你可以理解为结构体指针)(PIMAGE)数组存储图片(简单说就是结构体指针数组)；

使用 RectButton 结构体数组存储按钮；

使用 sys_edit 类数组存储输入框信息；

使用 RANK_INFO 结构体数组存储高分榜消息；

4. 算法

在数出周边旗帜数，打开周边格子，遇到空格时递归打开周边格子，判断点击的按钮id等中，使用了枚举算法；

在判断点击的按钮id时，使用了查找算法；

在游戏存档与高分榜存档的相关操作中，使用了文件操作；

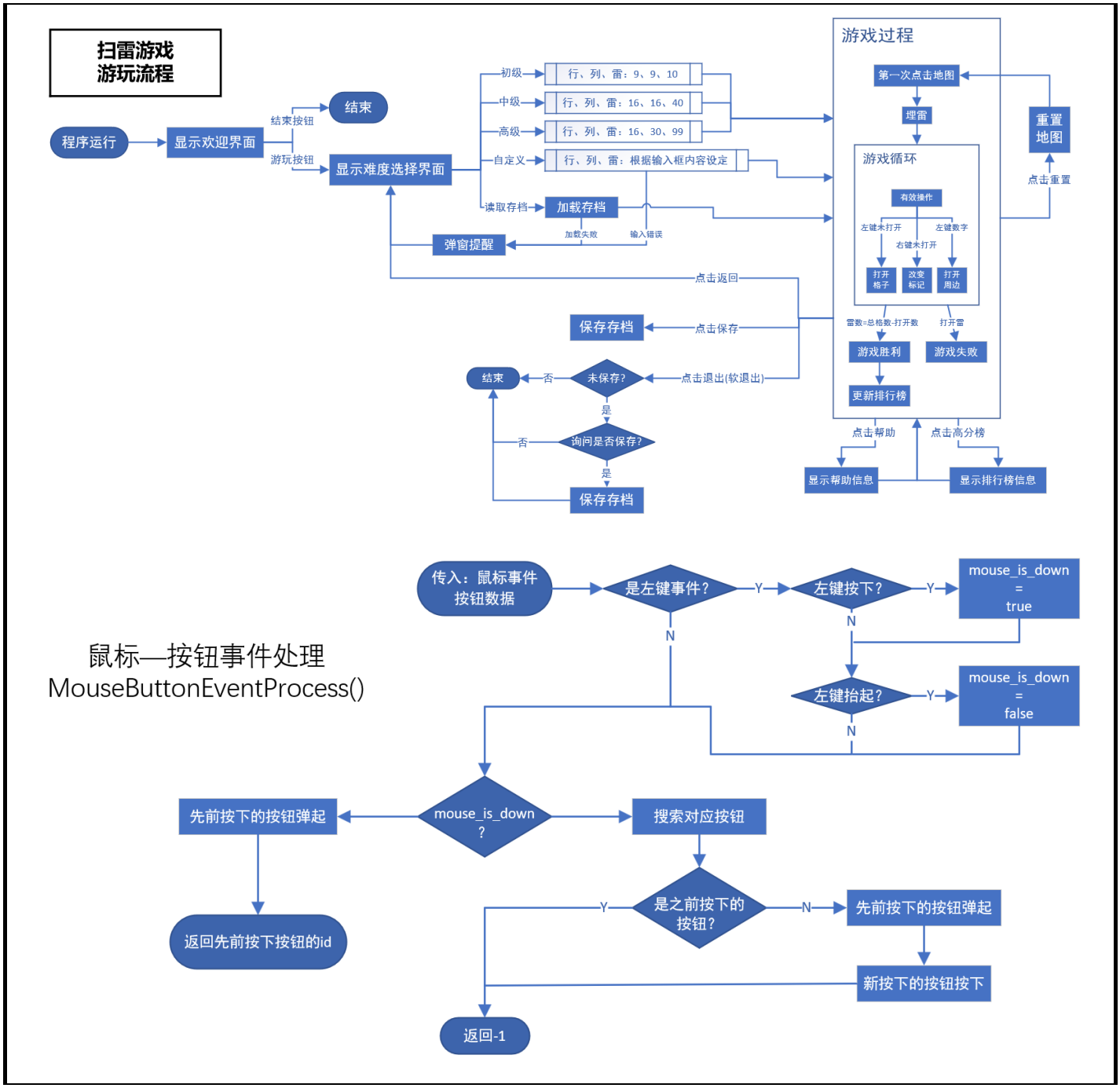
在埋雷，以及加密格子信息时，使用了随机算法；

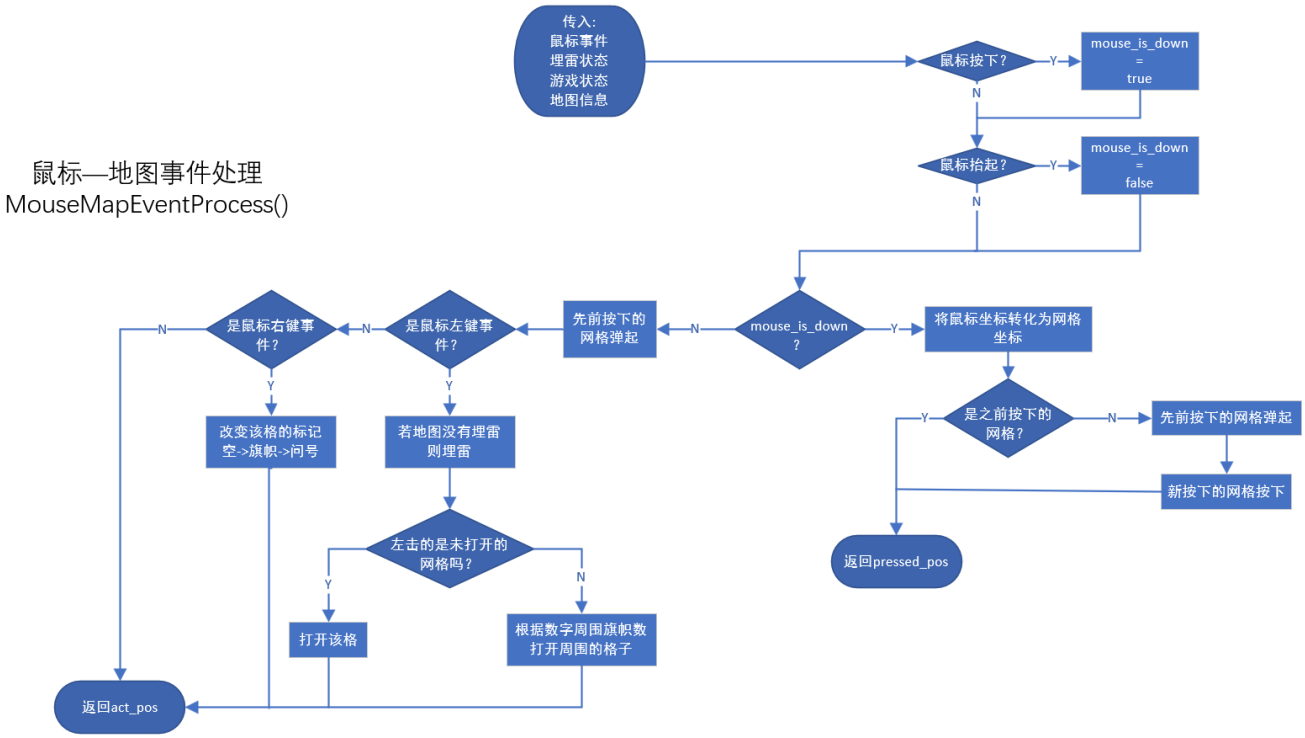
在遇到空格时递归打开周边格子时，使用了递归算法；

在保存存档及加载存档时，使用了简单的加密解密算法，通过加入随机位，异或，循环移位实现。

5. 程序流程图

游戏总体流程图如下：


扫雷游戏
游玩流程
程序运行
显示欢迎界面
结束按钮
结束
游玩按钮
显示难度选择界面
初级
行、列、雷：9、9、10
中级
行、列、雷：16、16、40
高级
行、列、雷：16、30、99
自定义
行、列、雷：根据输入框内容设定
读取存档
加载存档
加载失败
输入错误
弹窗提醒
游戏过程
第一次点击地图
埋雷
游戏循环
有效操作
左键未打开
左键数字
右键未打开
打开格子
改变标记
打开周边
重置地图
点击重置
雷数=总格数-打开数
打开雷
游戏胜利
游戏失败
更新排行榜
点击返回
保存存档
点击保存
结束
否
未保存?
点击退出(软退出)
是
否
询问是否保存?
是
保存存档
点击帮助
点击高分榜
显示帮助信息
显示排行榜信息
鼠标—按钮事件处理
MouseButtonEventProcess()
传入：鼠标事件
按钮数据
是左键事件?
Y
N
左键按下?
Y
N
mouse_is_down
=
true
左键抬起?
Y
N
mouse_is_down
=
false
mouse_is_down
?
先前按下的按钮弹起
返回先前按下按钮的id
搜索对应按钮
是之前按下的
按钮?
Y
N
先前按下的按钮弹起
新按下的按钮按下
返回-1



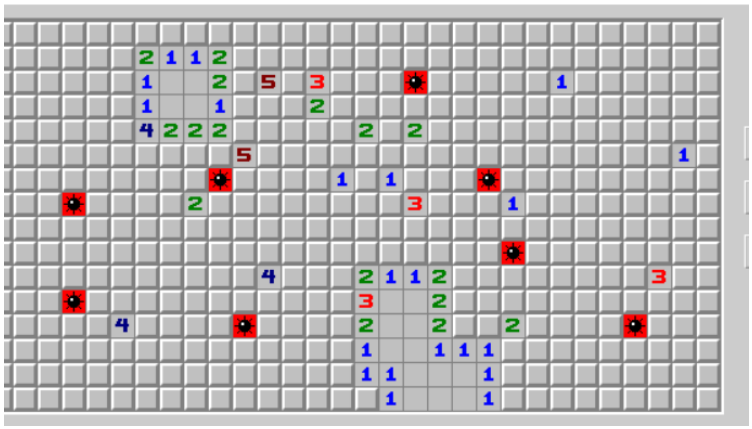
实验过程中遇到的问题及解决方法与思路：

问题 1： 编译出的 exe 文件无法被其他系同学打开，显示缺失部分文件

原因：该问题只会出现在没有在电脑中安装 C 语言有关内容时才会出现，在编写程序时也要考虑到程序在完全没有编程相关的电脑上能否执行

解决方法：在编译时加入-static 参数，使用静态链接库

问题 2： 在鼠标点击速度极快时，可做到点开雷但游戏不结束（匪夷所思的 bug）



原因：该问题是由“打开格子”操作与“判断失败”操作的异步进行引起的

解决方法：将“判断失败”操作改成在“打开格子”后立刻执行

问题 3：在其他电脑上出现存档打不开的情况

原因：打开文件时使用了绝对路径，在自己电脑可以运行，但在其他电脑上路径就不符预期了

解决方法：修改为相对路径

问题 4：游戏时发生莫名其妙的闪退
原因：数组越界
解决方法：在对数组进行操作时先判断操作是否合法

问题 5：在其他电脑编译时出现[Error] ld returned 1 exit status
原因：使用了与 C++关键字重复的 map 变量
解决方法：将 map 变量名改为 board 即可

问题 6：加密算法中的循环移位代码没有得到预期的结果
原因：C 语言中，对 int 类型，左移位是右补 0，但右移位时左补的是符号位，本人想当然的认为右移位也是左补 0
解决方法：把有关移位运算的 int 类型都改为 unsigned

问题 7：存档没有达到预期的随机效果
原因：在每次生成随机数前，都调用了 srand(time(0))，导致生成的随机数都是定值
解决方法：只调用一次 srand(time(0))

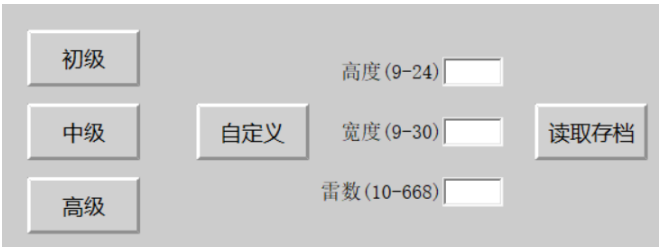
测试用例和系统测试结果：

测试按钮的功能：
用例 1：使用鼠标右键、中键单击按钮，并松开按键
结果：毫无反应

用例 2：使用鼠标左键单击按钮，但不松开按键
结果：按钮出现按下的动画，但没有执行相应指令

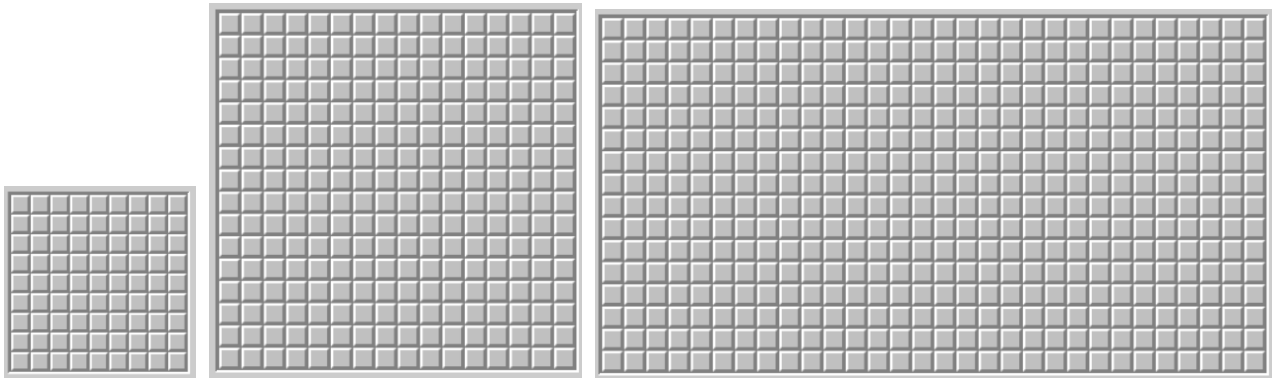
用例 3：在用例 2 的前提下，继续按住按键，将鼠标指针移开按钮范围，松开
结果：按钮弹起，仍没有执行相应指令

用例 4.：使用鼠标左键单击按钮，并在鼠标指针处于按钮范围时松开按键
结果：按钮弹起，执行了相应指令（弹起的瞬间执行了指令，导致难以看到按钮弹起）

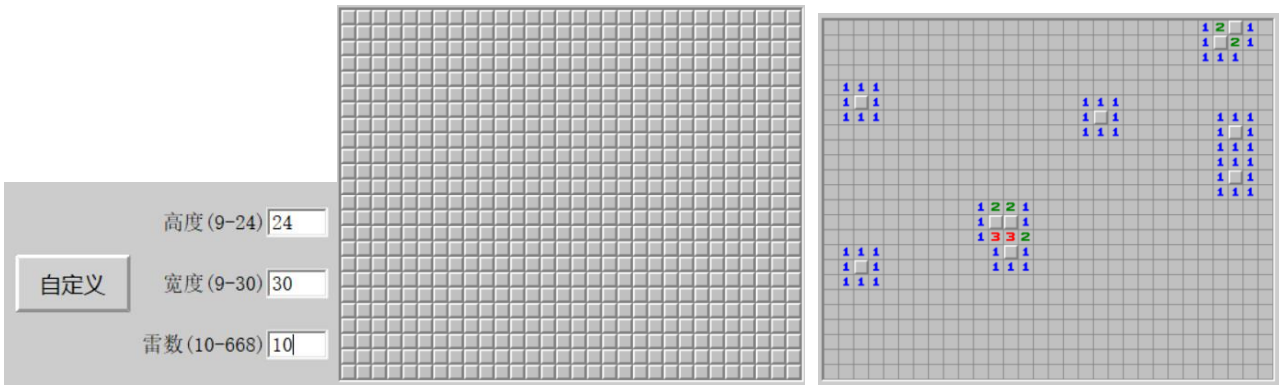


结论：按钮的反应行为完全符合预期

测试选择难度的功能：
用例 1：分别点击难度选择菜单中的初级、中级、高级按钮
结果：正确的设定了游戏参数



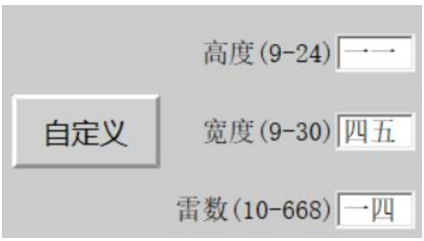
用例 2：在自定义难度的输入框中填入适当的数字（24 30 10），并点击自定义按钮
结果：游戏参数符合预期

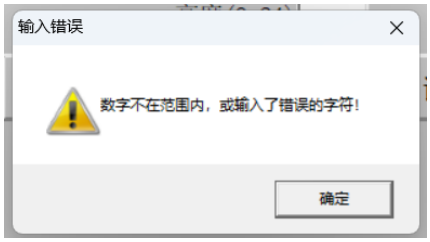


用例 3：在自定义难度的输入框中填入不适当的超出范围的数字（30 30 300），并点击自定义按钮
结果：出现弹窗提醒，输入框被清空

用例 4：在自定义难度的输入框中填入不适当的不符合实际意义的数字（10 10 300），并点击自定义按钮
结果：出现弹窗提醒，输入框被清空

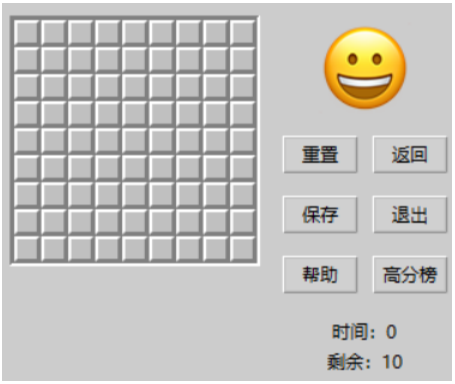
用例 5：在自定义难度的输入框中填入非法字符，并点击自定义按钮
结果：出现弹窗提醒，输入框被清空



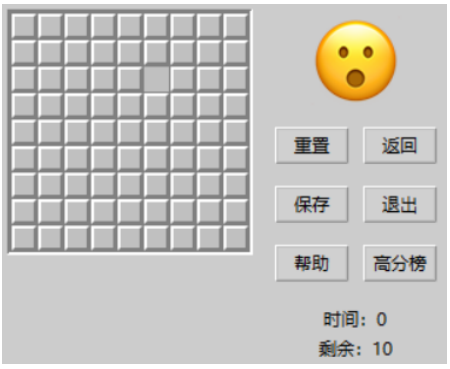


结论：难度选择菜单的功能符合预期，并具有处理非法输入的能力

测试基础游戏过程：（以初级为例）
用例 1：什么都不做（不点击第一次）
结果：游戏界面保持不变



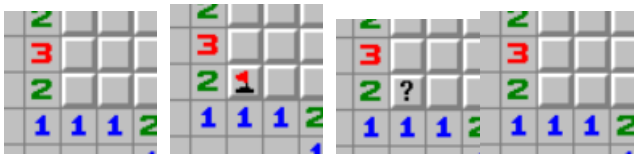
用例 2：鼠标分别单击（左、中、右键），但不松开
结果：被操作的格子有按下的动画，emoji 变成惊讶脸，但没有开始计时



用例 3：在用例 2 的基础上，保持鼠标按键按住，将鼠标指针移开格子范围
结果：在鼠标指针移动的路径上，经过的格子均有按下的动画，emoji 保持惊讶脸，仍没有开始计时；在鼠标移开格子范围后，没有格子被按下，emoji 变回笑脸，仍没有开始计时
用例 4：鼠标左键单击格子并松开
结果：格子被打开，计时开始

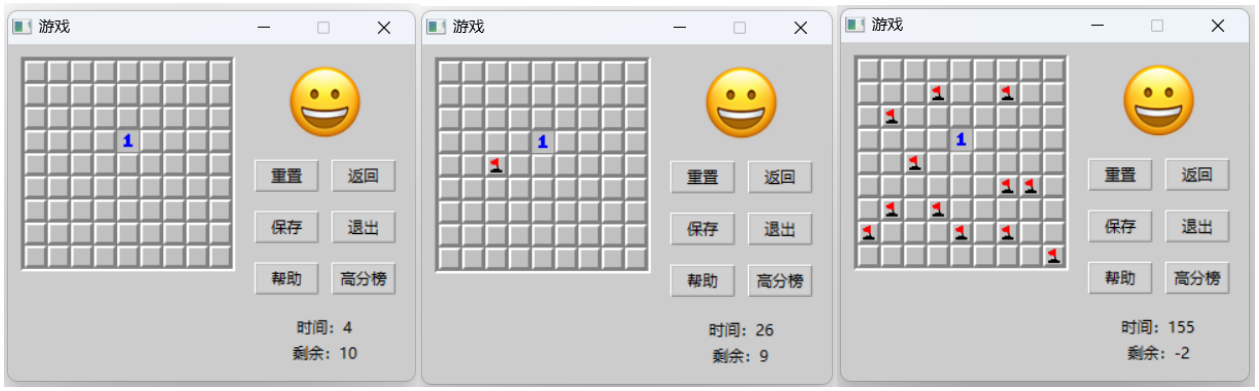
用例 5：右键单击格子

结果：操作格展示的状态从无→旗帜→问号→循环切换



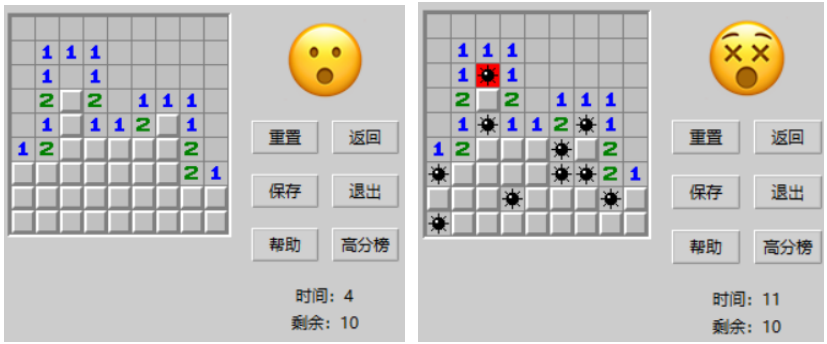
用例 6：右键标记随机格为旗帜

结果：右下角的雷数随着标记旗帜的增加而减少，符合预期（剩余雷数是可以为负的）



用例 7：左键单击雷格

结果：点击的格子展示为血雷格，其他的雷展示为普通雷格，emoji 变为死亡脸，计时结束，播放了死亡音效（由于无法做到在短时间内截多次屏，提供的案例图片在时间上有所偏差）



用例 8：完成一次游戏

结果：在点开最后一个非雷格后，播放了胜利音效，emoji 变为墨镜脸，计时结束

结论：游戏的简单功能与预期相符合

测试高级游戏过程：（以中级为例）

用例 1：在保证正确性的前提下，点击一个数字格子，其周围旗帜数与自身数字相同

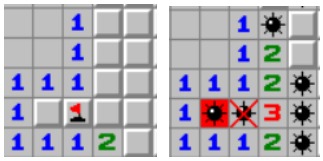
结果：打开了这个数字格周边的格子

用例 2：点击一个数字格子，其周围旗帜数与自身数字不相同（图中央的 3 数字格）

结果：未执行操作

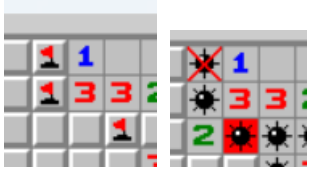
用例 3：在保证错误的前提下，点击一个数字格子（旗帜上方相邻的那个数字 1 格），其周围旗帜数与自身数字相同

结果：打开了这个数字格周边的格子，将正确的旗帜展示为普通雷格，将错误的旗帜展示为错误格，并将真正的雷展示为血雷格，其他与正常游戏结束流程一致



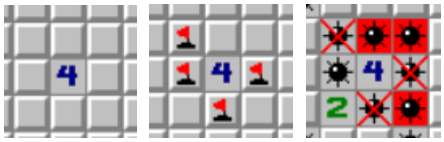
用例 4：较复杂的用例 3

结果：将正确的旗帜展示为普通雷格，其他表现与用例 3 结果一致

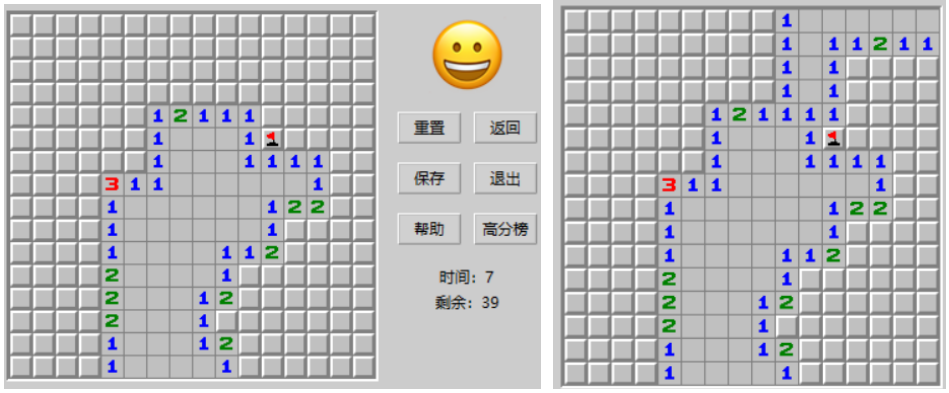


用例 5：更复杂的用例 3

结果：表现与用例 4 结果一致



用例 6：左键单击图中旗帜的左上角的数字 1
结果：打开了周边的格子，并且能正确的递归打开空格子

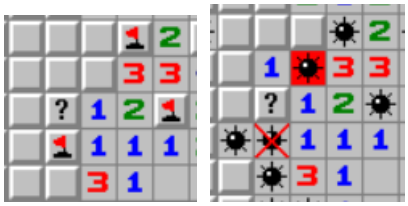


用例 7：左键单击问号下放的数字 1
结果：什么都没发生

用例 8：左键单击问号下放的数字 1
结果：什么都没发生

用例 8：左键单击问号左方的数字 3（这时问号是雷）
结果：打开了 3 周边的格子，但问号没有被打开，（第 3 张图为实际雷分布情况）

用例 9：左键单击问号右方的数字 1（这时问号不是雷）
结果：打开了 3 周边的格子，但问号没有被打开，且因为标雷错误导致游戏结束



结论：旗帜与问号的代码逻辑及行为与预期一致

测试存档的保存与读取：

用例 1：在游戏进行中时保存游戏

结果：弹窗提醒保存成功，并正确的输出了存档文件

用例 2：保存后，在难度选择界面点击“读取存档”按钮

结果：正确读取了存档，游戏界面与存档时保持一致

用例 3：人为修改存档文件，并在难度选择界面点击“读取存档”按钮

结果：弹窗提示存档损坏，仍留在难度选择界面

测试图片资源的读取：

用例：将 resourse 文件夹内的图片删去/修改文件名，并运行代码

结果：弹窗提醒图片读取错误，进程结束

程序的全部源代码：

main.cpp

```
#include "gui.h"
#include "calc.h"

PIMAGE button_img[2];//定义按钮图片数组。全局使用
BLOCK board[26][32];//定义地图，全局使用

int main(void)
{
    int command;
    MAP_INFO map_info;//定义地图信息结构体
    LoadButtonImage(button_img);

    command = ShowMenu();//显示菜单
    if (command == 1)//用户单击退出按钮
        exit(0);

    do{
        int game_type = GetInfo(&map_info);//通过图形界面，获取游戏信息
        command = GameStart(&map_info, game_type);//游戏开始
    }while (command == 1);

    ReleaseImage(button_img, 2);
    return 0;
}
```

calc.h

```
#pragma once

#include <string.h>
#include <stdlib.h>
#include <time.h>
#include <algorithm>
#include <graphics.h>

typedef struct MAP_INFO{//定义地图信息结构体
    int map_row, map_col, map_mines, map_flag, map_open, run_time;//地图的行数，列数，雷数，标旗数，以及打开的格子数，与已运行时间
}MAP_INFO;

typedef struct BLOCK{//地图中一格的数据结构体
    unsigned char num:4    ;//0~8 表示周围的雷数，15 表示本身就是雷
    unsigned char is_down:1;//该格是否被按下
    unsigned char is_open:1;//该格是否被打开
    unsigned char state:2  ;//该格的标记状态(0 为无标记，1 为标旗，2 为问号)
}BLOCK;//通过指定成员的位数，可大大减少地图存储所占的空间
```

```
19.
20. typedef struct MAP_POS{//定义地图坐标结构体
21.     int row;//地图坐标的行值，从1开始
22.     int col;//地图坐标的列值，从1开始
23.
24.     MAP_POS(int row=0, int col=0){//返回输入x，y值对应的地图坐标结构体
25.         this->row = row;
26.         this->col = col;
27.     }
28.
29.     MAP_POS operator+ (const MAP_POS& pos){//定义地图坐标之间的加法
30.         return MAP_POS(row + pos.row, col + pos.col);
31.     }
32.
33.     bool operator== (const MAP_POS& pos){//定义地图坐标之间的等于比较
34.         return row == pos.row && col == pos.col;
35.     }
36.
37.     bool operator!= (const MAP_POS& pos){//定义地图坐标之间的不等于比较
38.         return row != pos.row || col != pos.col;
39.     }
40.
41.     bool is_legal(const MAP_INFO* pmap_info) {//判断地图坐标是否合法
42.         return row >= 1 && row <= pmap_info->map_row && col >= 1 && col <= pmap_info->map_col;
43.     }
44. }MAP_POS;
45.
46. typedef struct RANK_INFO{//定义排名信息结构体
47.     char username[33];
48.     int time;
49. }RANK_INFO;
50.
51. //=================================================
52. //函数名：MouseToPos
53. //功能：将鼠标对窗口的相对坐标转换为地图坐标
54. //输入参数：鼠标信息(mouse_msg)
55. //返回值：对应的地图坐标(MAP_POS)
56. //=================================================
57. MAP_POS MouseToPos(mouse_msg msg);
58.
59. //=================================================
60. //函数名：PosToMouse
61. //功能：将地图坐标转换为鼠标对窗口的相对坐标(地图坐标对应格的左上角的像素坐标)
62. //输入参数：对应的地图坐标(MAP_POS)，用于接受鼠标相对于窗口坐标的指针(&x, &y)(int*,int*)
63. //返回值：无
64. //=================================================
65. void PosToMouse(MAP_POS pos, int* pmouse_x, int* pmouse_y);
66.
67. //=================================================
```

```
68. //函数名：InitialMap
69. //功能：初始化地图，将地图清空
70. //输入参数：无
71. //返回值：无
72. //=====================================================
73. void InitialMap(void);
74.
75. //=====================================================
76. //函数名：SetMines
77. //功能：埋雷，并更新雷周边的数字
78. //输入参数：用户点击的地图坐标(MAP_POS)，地图信息结构体指针(MAP_INFO*)
79. //返回值：无
80. //=====================================================
81. void SetMines(MAP_POS act_pos, const MAP_INFO* pmap_info);//设置地雷，并更新周围的数字
82.
83. //=====================================================
84. //函数名：OpenBlock
85. //功能：打开格子，若为空格，递归打开周边格子
86. //输入参数：用户点击的地图坐标(MAP_POS)，指向游戏状态的指针，指向地图信息结构体的结构体指针
87. //返回值：无
88. //=====================================================
89. void OpenBlock(MAP_POS act_pos, int* game_state, MAP_INFO* pmap_info);
90.
91. //=====================================================
92. //函数名：ChangeState
93. //功能：改变格子标记状态，按无标记->旗帜->问号循环切换
94. //输入参数：用户点击的地图坐标对应的格子的地址(BLOCK*)，地图旗帜数地址(int*)
95. //返回值：无
96. //=====================================================
97. void ChangeState(BLOCK* pblock, int* map_flag);//按无标记->旗帜->问号循环标记格子
98.
99. //=====================================================
100. //函数名：OpenSurround
101. //功能：打开周边的格子，当且仅当周边的旗帜数等于该格的数字
102. //输入参数：用户点击的地图坐标(MAP_POS)，游戏状态指针(int*)，地图信息结构体指针(MAP_INFO*)
103. //返回值：无
104. //=====================================================
105. void OpenSurround(MAP_POS act_pos, int* game_state, MAP_INFO* pmap_info);
106.
107. //=====================================================
108. //函数名：CountFlags
109. //功能：数出周围的旗帜数
110. //输入参数：用户点击的地图坐标(MAP_POS)
111. //返回值：周围的旗帜数(int)
112. //=====================================================
113. int CountFlags(MAP_POS act_pos);//计算周围的旗帜数
114.
115. //=====================================================
116. //函数名：EncryptBlock
```

```
117. //功能：将地图格子的信息进行加密
118. //输入参数：地图格子(BLOCK)
119. //返回值：加密结果(unsigned)
120. //=================================================
121. unsigned EncryptBlock(BLOCK block);
122.
123. //=================================================
124. //函数名：EncryptNum
125. //功能：将数字进行加密
126. //输入参数：数字(unsigned)
127. //返回值：加密结果(unsigned)
128. //=================================================
129. unsigned EncryptNum(unsigned data);
130.
131. //=================================================
132. //函数名：DecryptBlock
133. //功能：将地图格子的信息进行解密
134. //输入参数：加密结果(unsigned)
135. //返回值：解密结果(BLOCK)
136. //=================================================
137. BLOCK DecryptBlock(unsigned key);
138.
139. //=================================================
140. //函数名：DecryptNum
141. //功能：将数字进行解密
142. //输入参数：数字(unsigned)
143. //返回值：解密结果(unsigned)
144. //=================================================
145. unsigned DecryptNum(unsigned key);
```

calc.cpp

```
1.  #include "calc.h"
2.  extern BLOCK board[26][32];//引用外部变量
3.
4.  MAP_POS surround[8]={{-1, -1}, {-1, 0}, {-1, 1}, {0, -1}, {0, 1}, {1, -1}, {1, 0}, {1, 1}};//八个
    方向的移动
5.
6.  //=================================================
7.  //函数名：MouseToPos
8.  //功能：将鼠标对窗口的相对坐标转换为地图坐标
9.  //输入参数：鼠标信息(mouse_msg)
10. //返回值：对应的地图坐标(MAP_POS)
11. //=================================================
12. MAP_POS MouseToPos(mouse_msg msg)
13. {
14.     return MAP_POS((msg.y-12)/18 + 1, (msg.x-12)/18 + 1);
15. }
16.
17. //=================================================
```

```
18. //函数名：PosToMouse
19. //功能：将地图坐标转换为鼠标对窗口的相对坐标(地图坐标对应格的左上角的像素坐标)
20. //输入参数：对应的地图坐标(MAP_POS)，用于接受鼠标相对于窗口坐标的指针(&x, &y)(int*)
21. //返回值：无
22. //=================================================
23. void PosToMouse(MAP_POS pos, int* pmouse_x, int* pmouse_y)
24. {
25.     *pmouse_x = (pos.col-1)*18 + 12 + 1;
26.     *pmouse_y = (pos.row-1)*18 + 12 + 1;
27. }
28.
29. //=================================================
30. //函数名：InitialMap
31. //功能：初始化地图，将地图清空
32. //输入参数：无
33. //返回值：无
34. //=================================================
35. void InitialMap(void)//初始化地图
36. {
37.     memset(board, 0, sizeof(BLOCK)*26*32);
38. }
39.
40. //=================================================
41. //函数名：SetMines
42. //功能：埋雷，并更新雷周边的数字
43. //输入参数：用户点击的地图坐标(MAP_POS)，指向地图信息结构体的结构体指针(MAP_INFO*)
44. //返回值：无
45. //=================================================
46. void SetMines(MAP_POS act_pos, const MAP_INFO* pmap_info)
47. {
48.     srand(time(0));//设置随机数种子
49.     int mine_set=0;
50.
51.     do{
52.         int rand_row = rand() % pmap_info->map_row + 1;
53.         int rand_col = rand() % pmap_info->map_col + 1;
54.         if (board[rand_row][rand_col].num != 15 && MAP_POS(rand_row, rand_col) != act_pos){// 该格
    不为雷且不是用户点击的格子时
55.             board[rand_row][rand_col].num = 15;//埋雷
56.             mine_set++;
57.
58.             for (int i=0; i<8; i++){//对周围八个格子
59.                 if ((MAP_POS(rand_row, rand_col) + surround[i]).is_legal(pmap_info) &&\
60.                 board[rand_row + surround[i].row][rand_col + surround[i].col].num != 15){// 当格子
    坐标合法，且不是雷时
61.                     board[rand_row + surround[i].row][rand_col + surround[i].col].num++;//该格数字
    加1
62.                 }
63.             }
```

```
64.         }
65.     }while (mine_set < pmap_info->map_mines);//直到埋雷数等于地图雷数
66. }
67.
68. //=================================================
69. //函数名：OpenBlock
70. //功能：打开格子，若为空格，递归打开周边格子
71. //输入参数：用户点击的地图坐标(MAP_POS)，指向游戏状态的指针，指向地图信息结构体的结构体指针
72. //返回值：无
73. //=================================================
74. void OpenBlock(MAP_POS act_pos, int* game_state, MAP_INFO* pmap_info)
75. {
76.     if (act_pos.is_legal(pmap_info)){//当用户点击坐标合法时
77.         BLOCK* pblock = &board[act_pos.row][act_pos.col];//取个别名，方便使用
78.
79.         if (pblock->is_open == 0 && pblock->state == 0){//当且仅当格子没被打开，且格子没有被标记时，格
    子才能被打开
80.             pblock->is_open = 1;
81.             (pmap_info->map_open)++;//被打开的格子数加1
82.
83.             if (pblock->num == 15)//打开了雷
84.                 *game_state = 1;//游戏失败
85.
86.             if (pblock->num == 0)//当周边雷数为0时
87.                 for (int i=0; i<=7; i++){//遍历周边格子
88.                     if ((act_pos + surround[i]).is_legal(pmap_info))//当周边格子坐标合法时
89.                         OpenBlock(act_pos + surround[i], game_state, pmap_info);//递归打开周边的格
    子
90.                 }
91.         }
92.     }
93. }
94.
95. //=================================================
96. //函数名：ChangeState
97. //功能：改变格子标记状态
98. //输入参数：用户点击的地图坐标对应的格子的地址(BLOCK*)，地图中的旗帜数
99. //返回值：无
100. //=================================================
101. void ChangeState(BLOCK* pblock, int* map_flag)
102. {
103.     if (pblock->is_open  == 0){//格子没被打开，才能标记
104.         switch (pblock->state){
105.         case 0:
106.             pblock->state = 1;//普通格变为旗帜
107.             (*map_flag)++;//旗帜数加1
108.             break;
109.         case 1:
110.             pblock->state = 2;//旗帜变为问号
```

```
111.             (*map_flag)--;//旗帜数减1
112.             break;
113.         case 2:
114.             pblock->state = 0;//问号变为普通格
115.             break;
116.         }
117.     }
118. }
119.
120. //================================================
121. //函数名：OpenSurround
122. //功能：打开周边的格子，当且仅当周边的旗帜数等于该格的数字
123. //输入参数：用户点击的地图坐标(MAP_POS)，指向游戏状态的指针，指向地图信息结构体的结构体指针
124. //返回值：无
125. //================================================
126. void OpenSurround(MAP_POS act_pos, int* game_state, MAP_INFO* pmap_info)
127. {
128.     if (act_pos.is_legal(pmap_info)){
129.         BLOCK* pblock = &board[act_pos.row][act_pos.col];
130.
131.         if (pblock->is_open && pblock->num != 0 && pblock->num == CountFlags(act_pos))//当且仅当
    格子被打开时且周边的旗帜数等于该格的数字
132.             for (int i=0; i<=7; i++)//遍历周边格子
133.                 if ((act_pos + surround[i]).is_legal(pmap_info)){//当周边格子坐标合法时
134.                     OpenBlock(act_pos + surround[i], game_state, pmap_info);//打开周边格子
135.                 }
136.     }
137. }
138.
139. //================================================
140. //函数名：CountFlags
141. //功能：数出周围的旗帜数
142. //输入参数：用户点击的地图坐标(MAP_POS)
143. //返回值：周围的旗帜数(int)
144. //================================================
145. int CountFlags(MAP_POS act_pos)
146. {
147.     int flag_count=0;
148.     for (int i=0; i<8; i++)
149.         if (board[act_pos.row + surround[i].row][act_pos.col + surround[i].col].state == 1)
150.             flag_count++;
151.     return flag_count;
152. }
153.
154. //================================================
155. //函数名：EncryptBlock
156. //功能：将地图格子的信息进行加密
157. //输入参数：地图格子(BLOCK)
158. //返回值：加密结果(unsigned)
```

```
159. //=================================================
160. unsigned EncryptBlock(BLOCK block)
161. {
162.     unsigned num =((rand()%8388608)<<8) + (block.num<<4) + (block.is_down<<3) + (block.is_open<<
    2) + block.state;//前24位为随机数，后8位存放信息
163.     num ^= 1571530884;//加掩码
164.     num = num>>23 | num<<(32-23);//循环移位
165.     num ^= 2145043451;//加掩码
166.     return num;
167. }
168.
169. //=================================================
170. //函数名：EncryptNum
171. //功能：将数字进行加密
172. //输入参数：数字(unsigned)
173. //返回值：加密结果(unsigned)
174. //=================================================
175. unsigned EncryptNum(unsigned data)
176. {
177.     data ^= 1839588510;
178.     data = data>>19 | data<<(32-19);
179.     data ^= 1977583954;
180.     return data;
181. }
182.
183. //=================================================
184. //函数名：DecryptBlock
185. //功能：将地图格子的信息进行解密
186. //输入参数：加密结果(unsigned)
187. //返回值：解密结果(BLOCK)
188. //=================================================
189. BLOCK DecryptBlock(unsigned key)
190. {
191.     key ^= 2145043451;
192.     key = key<<23 | key>>(32-23);
193.     key ^= 1571530884;
194.     BLOCK block;
195.     block.num = (key&240)>>4;   //240=11110000
196.     block.is_down = (key&8)>>3; //8=00001000
197.     block.is_open = (key&4)>>2; //4=00000100
198.     block.state = key&3;        //3=00000011
199.     return block;
200. }
201.
202. //=================================================
203. //函数名：DecryptNum
204. //功能：将数字进行解密
205. //输入参数：数字(unsigned)
206. //返回值：解密结果(unsigned)
```

```
207. //================================================
208. unsigned DecryptNum(unsigned key)
209. {
210.     key ^= 1977583954;
211.     key = key<<19 | key>>(32-19);
212.     key ^= 1839588510;
213.     return key;
214. }
```

gui.h

```
1.  #pragma once
2.  #define MENU_WIDTH 640
3.  #define MENU_HEIGHT 400
4.  #define SELECT_WIDTH 620
5.  #define SELECT_HEIGHT 240
6.
7.  #include "load.h"
8.  #include "calc.h"
9.
10. typedef struct{
11.     int x, y;//按钮左上角像素的坐标
12.     int width, height;//按钮的宽与长
13.     bool is_press;//按钮是否被按下
14.     const char* label;//按钮的标签
15. }RectButton;
16.
17. //================================================
18. //函数名：ShowMenu
19. //功能：显示菜单，及其按钮与对应功能
20. //输入参数：无
21. //返回值：用户触发按钮的id
22. //================================================
23. int ShowMenu(void);
24.
25. //================================================
26. //函数名：GetInfo
27. //功能：显示选择菜单，及其按钮与对应功能，主要功能是获取游戏地图的数据
28. //输入参数：指向地图信息结构体的结构体指针(MAP_INFO*)
29. //返回值：0 为开启新游戏，1 为继续游戏
30. //================================================
31. int GetInfo(MAP_INFO* pmap_info);
32.
33. //================================================
34. //函数名：GameStart
35. //功能：包括扫雷游戏的主要内容
36. //输入参数：指向地图信息结构体的结构体指针(MAP_INFO*)，游戏类型(int，0 为开启新游戏；1 为继续上局游戏，需读
    取文件)
37. //返回值：0 为退出，1 为返回难度设置界面
38. //================================================
```

```
39. int GameStart(MAP_INFO* pmap_info, int game_type);
40.
41. //=================================================
42. //函数名：DrawButton
43. //功能：按指定字体大小绘制所有按钮
44. //输入参数：按钮数组(RectButton*)，按钮数量(int)，字体大小(int)
45. //返回值：无
46. //=================================================
47. void DrawButton(const RectButton* button, int n, int font_size);
48.
49. //=================================================
50. //函数名：SearchButton
51. //功能：搜索输入坐标所对应的按钮 id
52. //输入参数：坐标 x，y，按钮数组(RectButton*)，按钮数量(int)
53. //返回值：输入坐标所对应的按钮 id，-1 为无对应按钮
54. //=================================================
55. int SearchButton(int x, int y, const RectButton* button, int n);
56.
57. //=================================================
58. //函数名：IsInBottonRange
59. //功能：判断坐标是否在按钮范围内
60. //输入参数：需要判断的坐标(int,int)(x，y 值)，与需要判断的按钮(RectButton*)
61. //返回值：bool 类型，表明坐标是否在按钮范围内
62. //=================================================
63. bool IsInBottonRange(int x, int y, const RectButton* button);
64.
65. //=================================================
66. //函数名：MouseButtonEventProcess
67. //功能：处理鼠标事件，反映在按钮上
68. //输入参数：鼠标消息，按钮数组(RectButton*)，按钮数量(int)
69. //返回值：用户选择的按钮 id，-1 为未选择
70. //=================================================
71. int MouseButtonEventProcess(mouse_msg msg, RectButton* button, int n);
72.
73. //=================================================
74. //函数名：MouseMapEventProcess
75. //功能：处理鼠标事件，反映在地图上
76. //输入参数：鼠标消息(mouse_msg)，雷是否已埋下(bool)，游戏状态指针(int*)，地图信息结构体指针(MAP_INFO*)
77. //返回值：鼠标消息中鼠标位置对应的地图坐标
78. //=================================================
79. MAP_POS MouseMapEventProcess(mouse_msg msg, bool* map_is_set, int* game_state, MAP_INFO* pmap_inf
    o);
80.
81. //=================================================
82. //函数名：DrawBoundary
83. //功能：绘制网格
84. //输入参数：地图信息结构体指针(MAP_INFO*)
85. //返回值：无
86. //=================================================
```

```
87. void DrawBoundary(MAP_INFO* pmap_info);
88.
89. //=================================================
90. //函数名：DrawImage
91. //功能：绘制网格图片
92. //输入参数：游戏图片数组(PIMAGE*)，地图信息结构体指针(int*)，游戏运行状态(int)
93. //返回值：无
94. //=================================================
95. void DrawImage(PIMAGE* game_img, MAP_INFO* pmap_info, int game_state);
96.
97. //=================================================
98. //函数名：DrawEmoji
99. //功能：绘制Emoji图片
100. //输入参数：游戏图片数组(PIMAGE*)，是否展示惊讶的布尔值(bool)，游戏运行状态(int)，地图的列数(int)
101. //返回值：无
102. //=================================================
103. void DrawEmoji(PIMAGE* game_img, bool is_surprised, int game_state, int map_col);
```

gui.cpp

```
1.  #include "gui.h"
2.  #include "load.h"
3.  #include "calc.h"
4.  #include <ege/sys_edit.h>
5.
6.  extern BLOCK board[26][32];//引用外部变量
7.
8.  //=================================================
9.  //函数名：ShowMenu
10. //功能：显示菜单，及其按钮与对应功能
11. //输入参数：无
12. //返回值：用户触发按钮的id(int)
13. //=================================================
14. int ShowMenu(void)
15. {
16.     initgraph(MENU_WIDTH, MENU_HEIGHT, INIT_RENDERMANUAL | INIT_WITHLOGO);//初始化菜单窗口，为手动刷
    新模式
17.     setcaption("菜单");//设置标题
18.
19.     PIMAGE menu_background = newimage();//创建并加载菜单背景图片
20.     getimage(menu_background, "resource\\menu_background.png");
21.
22.     RectButton menu_button[2]={{88       , MENU_HEIGHT-45-100, 210, 100, false, "游 玩"},
23.                                {88+210+44, MENU_HEIGHT-45-100, 210, 100, false, "退 出"}};
24.     //创建按钮数组，0为游玩按钮，1为退出按钮
25.
26.     int command = -1;//存储用户点击的按钮，-1为未操作，0为游玩，1为退出
27.
28.     for (; command==-1; delay_fps(60)){//绘图循环
29.         mouse_msg msg;
```

```
30.         while (mousemsg()){
31.             msg = getmouse();
32.             command = MouseButtonEventProcess(msg, menu_button, 2);//处理鼠标事件
33.         }
34.
35.   cleardevice();//清屏
36.         putimage(0, 0, menu_background);//绘制背景图片
37.   DrawButton(menu_button, 2, 50);//绘制按钮
38.     }
39.     closegraph();//关闭菜单窗口
40.     ReleaseImage(&menu_background, 1);//释放图片内存
41.     return command;
42. }
43.
44. //=================================================
45. //函数名：GetInfo
46. //功能：显示选择菜单，及其按钮与对应功能，主要功能是获取游戏地图的数据
47. //输入参数：指向地图信息结构体的结构体指针
48. //返回值：0 为开启新游戏，1 为继续游戏
49. //=================================================
50. int GetInfo(MAP_INFO* pmap_info)
51. {
52.     initgraph(SELECT_WIDTH, SELECT_HEIGHT, INIT_RENDERMANUAL);//初始化窗口，为手动刷新模式
53.     setcaption("难度设置");//设置标题
54.     setbkcolor(EGERGB(204, 204, 204));//设置背景色
55.
56.     sys_edit input_box[3];
57.     for (int i=0; i<3; i++){
58.         input_box[i].create(false);
59.         input_box[i].move(408, 54+i*(26+27));
60.         input_box[i].size(18 * 3, 18 + 8);
61.         input_box[i].setmaxlen(3);
62.         input_box[i].setfont(18, 0, "宋体");
63.         input_box[i].visible(true);
64.     }
65.     //创建并初始化输入框数组，0 为输入地图行数，1 为输入地图列数，2 为输入地图雷数
66.
67.     RectButton select_button[5]={{40     , 30,           100, 50, false, "初级"},
68.                                  {40     , 30+50+15,     100, 50, false, "中级"},
69.                                  {40     , 30+50*2+15*2, 100, 50, false, "高级"},
70.                                  {240-50 , 30+50+15,     100, 50, false, "自定义"},
71.                                  {620-130, 30+50+15,     100, 50, false, "读取存档"}};
72.     //创建按钮数组，0 为初级按钮，1 为中级按钮，2 为高级按钮，3 为自定义按钮，4 为读取存档按钮
73.
74.     int command;//存储用户点击的按钮，-1 为未操作，0 为初级按钮，1 为中级按钮，2 为高级按钮，3 为自定义按钮，
    4 为读取存档按钮
75.     bool input_no_error, save_no_error;//记录自定义内容是否输入正确
76.     do{
77.         input_no_error = save_no_error = true;
```

```
78.         command = -1;
79.
80.         for (int i=0; i<3; i++)
81.             input_box[i].settext("");//清空输入框
82.
83.         for (; command==-1; delay_fps(60)){//绘图循环
84.             mouse_msg msg;
85.             while (mousemsg()){
86.                 msg = getmouse();//获取鼠标事件
87.                 command = MouseButtonEventProcess(msg, select_button, 5);//处理鼠标按钮事件
88.             }
89.
90.             cleardevice();//清屏
91.
92.             DrawButton(select_button, 5, 25);//绘制按钮
93.             setbkmode(TRANSPARENT);//设置文字背景透明
94.             setfont(18, 0, "宋体");//设置字体
95.             settextjustify(RIGHT_TEXT, CENTER_TEXT);//对齐模式：水平右对齐，垂直居中
96.             outtextxy(408, 54+0*(26+27)+13, "高度(9-24)");
97.             outtextxy(408, 54+1*(26+27)+13, "宽度(9-30)");
98.             outtextxy(408, 54+2*(26+27)+13, "雷数(10-668)");
99.         }
100.
101.         switch (command){//根据指令设置难度
102.         case 0://初级
103.             pmap_info->map_row = 9;
104.             pmap_info->map_col = 9;
105.             pmap_info->map_mines = 10;
106.             break;
107.         case 1://中级
108.             pmap_info->map_row = 16;
109.             pmap_info->map_col = 16;
110.             pmap_info->map_mines = 40;
111.             break;
112.         case 2://高级
113.             pmap_info->map_row = 16;
114.             pmap_info->map_col = 30;
115.             pmap_info->map_mines = 99;
116.             break;
117.         case 3://自定义
118.             char str_buffer[8];
119.             for (int i=0; i<3; i++){//获取三个输入框的内容
120.                 input_box[i].gettext(8, str_buffer);
121.                 int num;
122.
123.                 if (1 != sscanf(str_buffer, "%d", &num)){
124.                     input_no_error = false;
125.                     break;
126.                 }
```

```
127.
128.                 if (i==0){
129.                     if (num<9 || num>24){//行数超出范围
130.                         input_no_error = false;
131.                         break;
132.                     }else
133.                         pmap_info->map_row = num;
134.                 }else if (i==1){
135.                     if (num<9 || num>30){//列数超出范围
136.                         input_no_error = false;
137.                         break;
138.                     }else
139.                         pmap_info->map_col = num;
140.                 }else{
141.                     if (num<10 || num>668 || num >= pmap_info->map_row * pmap_info->map_col){//
    雷数超出范围
142.                         input_no_error = false;
143.                         break;
144.                     }else
145.                         pmap_info->map_mines = num;
146.                 }
147.             }
148.             break;
149.         case 4://继续上局
150.             save_no_error = !LoadSave(pmap_info, board);
151.             break;
152.         }
153.
154.         if (!input_no_error)//出错时，弹出提示框
155.             MessageBoxA(getHWnd(), "数字不在范围内，或输入了错误的字符！", "输入错误
    ", MB_ICONWARNING | MB_APPLMODAL);
156.     }while (!input_no_error || !save_no_error);
157.     closegraph();//关闭窗口
158.
159.     return command==4 ? 1 : 0;//0 为开启新游戏，1 为继续游戏
160. }
161.
162. //====================================================
163. //函数名：GameStart
164. //功能：包括扫雷游戏的主要内容
165. //输入参数：指向地图信息结构体的结构体指针，游戏类型(0 为开启新游戏；1 为继续上局游戏，需读取文件)
166. //返回值：0 为退出，1 为返回难度设置界面
167. //====================================================
168. int GameStart(MAP_INFO* pmap_info, int game_type)
169. {
170.     int game_width=10+2+18*pmap_info->map_col+3+140;
171.     int game_height=std::max(10+2+18*pmap_info->map_row+3+10, 263);
172.
173.     initgraph(game_width, game_height, INIT_RENDERMANUAL);//初始化菜单窗口，为手动刷新模式
```

```
174.     setcaption("游戏");//设置标题
175.     setbkcolor(EGERGB(204, 204, 204));//设置背景色
176.
177.     PIMAGE game_img[18];//创建并加载菜单背景图片
178.     LoadGameImage(game_img);
179.
180.     RectButton game_button[6]={{game_width-125,  90, 50, 25, false, "重置"},
181.                                {game_width-65 ,  90, 50, 25, false, "返回"},
182.                                {game_width-125, 130, 50, 25, false, "保存"},
183.                                {game_width-65 , 130, 50, 25, false, "退出"},
184.                                {game_width-125, 170, 50, 25, false, "帮助"},
185.                                {game_width-65 , 170, 50, 25, false, "高分榜"}};
186.     //创建按钮数组，0 为重试按钮，1 为返回难度选择按钮，2 为保存按钮，3 为退出按钮，4 为帮助按钮，5 为打开排行榜按钮
187.
188.     RANK_INFO rank_info[3];//定义排名信息数组
189.     LoadRank(rank_info);
190.
191.     int command;//存储用户点击的按钮，-1 为未操作，0 为重试，1 为返回，2 为保存，3 为退出，4 为帮助，5 为排行榜
192.     MAP_POS act_pos;//存储用户点击鼠标位置对应的地图坐标
193.     int game_state;//定义游戏运行状态，0 为游戏中，1 为死亡，2 为胜利
194.     bool mine_is_set;
195.
196.     do{//游戏循环
197.         command = -1;
198.         game_state = 0;
199.         int current_runtime = 0;
200.         bool mouse_is_down=false, dead_sound_is_played=false, game_is_saved=false;
201.         int start_time = -1, end_time;//记录游戏开始时间与结束时间
202.
203.         if (game_type == 0){//进行默认游戏
204.             mine_is_set = false;
205.             pmap_info->map_flag = pmap_info->map_open = pmap_info->run_time = 0;
206.             InitialMap();
207.         }else{//使用存档游戏
208.             mine_is_set = true;
209.             game_type = 0;
210.         }
211.         current_runtime = pmap_info->run_time;
212.         PlaySound(TEXT("resource\\start.wav"),nullptr,SND_ASYNC);
213.
214.         for (; command!=0 && command!=1 && command!=3; delay_fps(60)){//绘图循环
215.             command = -1;
216.
217.             mouse_msg msg;
218.             while (mousemsg()){
219.                 msg = getmouse();//获取鼠标事件
220.
```

```
221.                 // 判断鼠标按下与否
222.                 if (msg.is_down())
223.                     mouse_is_down = true;
224.                 if (msg.is_up())
225.                     mouse_is_down = false;
226.
227.                 command = MouseButtonEventProcess(msg, game_button, 6);//处理鼠标按钮事件
228.                 if (game_state == 0)
229.                     act_pos = MouseMapEventProcess(msg, &mine_is_set, &game_state, pmap_info);//
    处理鼠标网格事件
230.             }
231.
232.             if (start_time==-1 && mine_is_set)
233.                 start_time = time(0);//游戏开始后(用户点击、埋雷后)，记录开始时间
234.
235.             if (game_state == 0 && pmap_info->map_mines == pmap_info->map_row*pmap_info->map_col
     - pmap_info->map_open){
236.                 //游戏没有失败，且地图中剩余的格数等于雷数
237.                 game_state = 2;//游戏胜利
238.                 end_time = time(0);//记录游戏结束时间
239.                 PlaySound(TEXT("resource\\winner.wav"),nullptr,SND_ASYNC);
240.                 RankUpdate(rank_info, pmap_info, end_time-start_time);
241.             }
242.
243.             if (game_state == 1 && !dead_sound_is_played){//游戏失败后，且死亡音效没有播放
244.                 end_time = time(0);//记录游戏结束时间
245.                 PlaySound(TEXT("resource\\dead.wav"),nullptr,SND_ASYNC);//播放死亡音效
246.                 dead_sound_is_played = true;
247.             }
248.
249.             cleardevice();//清屏
250.             DrawBoundary(pmap_info);//绘制分界线
251.             DrawImage(game_img, pmap_info, game_state);//绘制格子图片
252.             DrawEmoji(game_img, act_pos.is_legal(pmap_info) && mouse_is_down, game_state, pmap_i
    nfo->map_col);//绘制emoji
253.             DrawButton(game_button, 6, 18);//绘制按钮
254.
255.             setbkmode(TRANSPARENT);//设置文字背景透明
256.             setcolor(BLACK);//设置字体颜色
257.             setfont(18, 0, "微软雅黑");//设置字体
258.             settextjustify(CENTER_TEXT, CENTER_TEXT);//文字以坐标为中心来显示
259.             xyprintf(game_width-70, 220, "                                                    时
    间：%d", mine_is_set ? ((game_state==0 ? time(0) : end_time) - start_time + current_runtime) : 0);
    //绘制所用时间
260.             xyprintf(game_width-70, 240, "                                                    剩
    余：%d", pmap_info->map_mines - pmap_info->map_flag);//绘制剩余雷数(实际雷数-旗帜数)
261.
262.             switch (command){
263.             case 2://选择了保存按钮
```

```
                if (game_state==0 && mine_is_set){//当游戏正在进行，且没有结束时，才能进行保存
                    pmap_info->run_time += time(0) - start_time;//写入游戏时间
                    SaveGame(pmap_info, board);
                    game_is_saved = true;
                }else
                    MessageBoxA(getHWnd(), "只能保存已经开始，且尚未结束的游戏！", "警告", MB_ICONWARNING | MB_APPLMODAL);
                break;
            case 3://选择了退出按钮
                if (game_state==0 && mine_is_set && !game_is_saved)
                    if (IDYES == MessageBoxA(getHWnd(), "游戏尚未保存，需要保存游戏吗？", "注意", MB_ICONQUESTION | MB_APPLMODAL | MB_YESNO)){
                        pmap_info->run_time += time(0) - start_time;//写入游戏时间
                        SaveGame(pmap_info, board);
                }
                break;
            case 4://选择了帮助按钮
                MessageBoxA(getHWnd(), "游戏操作:\n 鼠标左键单击未打开格子:\n"
                                       "\t 打开格子(不必担心第一步踩雷)\n"
                                       "鼠标左键单击数字:\n"
                                       "\t 当周边旗帜数等于数字时,打开周边格子\n"
                                       "鼠标右键单击未打开格子:\n"
                                       "\t 改变标记(无->旗帜->问号,循环)\n\n"
                                       "所有操作仅当按键抬起时才会操作(有后悔的机会)\n"
                                       "祝您游戏愉快!", "帮助", MB_ICONASTERISK | MB_APPLMODAL);
                break;
            case 5://选择了高分榜按钮
                char buffer[256];
                sprintf(buffer, "难度  用户名      \t 时间\n"
                                "初级  %-16s\t%d\n"
                                "中级  %-16s\t%d\n"
                                "高级  %-16s\t%d", \
                    rank_info[0].username, rank_info[0].time, \
                    rank_info[1].username, rank_info[1].time, \
                    rank_info[2].username, rank_info[2].time);
                MessageBoxA(getHWnd(), buffer, "高分榜", MB_APPLMODAL);
            }
        }
    }while (command == 0);//选择继续时，重新开始循环

    closegraph();//关闭菜单窗口
    ReleaseImage(game_img, 18);//释放图片内存
    return command;
}

//=================================================
//函数名：IsInBottonRange
//功能：判断坐标是否在按钮范围内
//输入参数：需要判断的坐标(x，y 值)(int, int)，与需要判断的按钮(RectButton*)
```

```
311. //返回值：(bool)表明坐标是否在按钮范围内
312. //================================================
313. bool IsInBottonRange(int x, int y, const RectButton* button)
314. {
315.     return (x >= button->x) && (x < button->x + button->width) && (y >= button->y) && (y < butto
    n->y + button->height);
316. }
317.
318. //================================================
319. //函数名：DrawButton
320. //功能：按指定字体大小绘制所有按钮
321. //输入参数：按钮数组(RectButton*)，按钮数量(int)，字体大小(int)
322. //返回值：无
323. //================================================
324. void DrawButton(const RectButton* button, int n, int font_size)
325. {
326.     extern PIMAGE button_img[2];
327.
328.     setbkmode(TRANSPARENT);//设置文字背景透明
329.     setcolor(BLACK);//设置字体颜色
330.     setfont(font_size, 0, "微软雅黑");//设置字体
331.     settextjustify(CENTER_TEXT, CENTER_TEXT);//文字以坐标为中心来显示
332.
333.     for (int i = 0; i < n; i++) {
334.         int x = button[i].x + button[i].width/2;//按钮中心的x 坐标
335.         int y = button[i].y + button[i].height/2;//按钮中心的y 坐标
336.
337.   if (button[i].is_press){
338.    putimage(button[i].x, button[i].y, button[i].width, button[i].height, button_img[1], 0, 0, 75
    9, 381);
339.    outtextxy(x, y+button[i].height/18, button[i].label);
340.             //绘制按下的按钮
341.   }else{
342.    putimage(button[i].x, button[i].y, button[i].width, button[i].height, button_img[0], 0, 0, 75
    9, 381);
343.    outtextxy(x, y, button[i].label);
344.             //绘制弹起的按钮
345.   }
346.  }
347. }
348.
349. //================================================
350. //函数名：SearchButton
351. //功能：搜索输入坐标所对应的按钮id
352. //输入参数：坐标x，y(int, int)，按钮数组(RectButton*)，按钮数量(int)
353. //返回值：输入坐标所对应的按钮id(int)，-1 为无对应按钮
354. //================================================
355. int SearchButton(int x, int y, const RectButton* button, int n)
356. {
```

```
357.  for (int i = 0; i < n; i++){
358.   if (IsInBottonRange(x, y, button + i)){
359.    return i;//退出，已经检测到，后面的按钮不再检测
360.   }
361.  }
362.  return -1;
363. }
364.
365. //=================================================
366. //函数名：MouseButtonEventProcess
367. //功能：处理鼠标事件，反映在按钮上
368. //输入参数：按钮数组(RectButton*)，按钮数量(int)
369. //返回值：用户选择的按钮id(int)，NONE(-1)为未选择
370. //=================================================
371. int MouseButtonEventProcess(mouse_msg msg, RectButton* button, int n)
372. {
373.     static bool mouse_is_down = false;
374.     static int pressed_button = NONE;
375.     int id = NONE;
376.
377.     // 判断鼠标左键按下与否
378.  if (msg.is_left()){
379.   if (msg.is_down())
380.             mouse_is_down = true;
381.         if (msg.is_up())
382.             mouse_is_down = false;
383.     }
384.     if (mouse_is_down){
385.         int select_button = SearchButton(msg.x, msg.y, button, n);//搜索鼠标按下时对应的按钮
386.
387.         if (select_button != pressed_button){//若不是先前按下的按钮
388.             if (pressed_button != NONE)
389.                 button[pressed_button].is_press = false;//先前按下的按钮弹起
390.             pressed_button = select_button;
391.             if (pressed_button != NONE)
392.                 button[pressed_button].is_press = true;//现在对应的按钮按下
393.         }
394.     }else{//鼠标弹起时，返回对应的按钮id值
395.         id = pressed_button;
396.         if (pressed_button != NONE)
397.             button[pressed_button].is_press = false;//按钮弹起
398.         pressed_button = NONE;//显然这时没有按钮被按下
399.     }
400.     return id;
401. }
402.
403. //=================================================
404. //函数名：MouseMapEventProcess
405. //功能：处理鼠标事件，反映在地图上
```

```
406. //输入参数：鼠标消息，雷是否已埋下，指向游戏状态的指针，指向地图信息结构体的结构体指针
407. //返回值：鼠标消息中鼠标位置对应的地图坐标(MAP_POS)
408. //==================================================
409. MAP_POS MouseMapEventProcess(mouse_msg msg, bool* map_is_set, int* game_state, MAP_INFO* pmap_in
    fo)
410. {
411.     static bool mouse_is_down = false;
412.     static MAP_POS pressed_pos(0, 0);
413.     static MAP_POS act_pos(0, 0);
414.
415.     // 判断鼠标按下与否
416.  if (msg.is_down())
417.         mouse_is_down = true;
418.     if (msg.is_up())
419.         mouse_is_down = false;
420.
421.     if (mouse_is_down){
422.         MAP_POS select_pos = MouseToPos(msg);//搜索鼠标按下时对应的地图坐标
423.
424.         if (select_pos != pressed_pos){//若不是先前按下的地图坐标
425.    if (pressed_pos.is_legal(pmap_info))
426.                 board[pressed_pos.row][pressed_pos.col].is_down = 0;//先前按下的地图坐标对应网格弹
    起
427.             pressed_pos = select_pos;
428.             if (pressed_pos.is_legal(pmap_info))
429.                 board[pressed_pos.row][pressed_pos.col].is_down = 1;//现在对应的地图坐标对应网格按
    下
430.         act_pos = pressed_pos;
431.   }
432.  }else{//鼠标弹起时，根据键位(左右)来执行操作，并返回操作的地图坐标
433.         act_pos = pressed_pos;
434.         if (pressed_pos.is_legal(pmap_info)){
435.             board[pressed_pos.row][pressed_pos.col].is_down = 0;//网格弹起
436.             pressed_pos = MAP_POS(0, 0);//显然这时没有网格被按下
437.
438.             if (msg.is_left()){//鼠标左键打开未打开的格子，或者打开周边格子，当且仅当周边旗帜数等于该格
    周边的雷数
439.                 if (!*map_is_set){
440.                     SetMines(act_pos, pmap_info);
441.                     *map_is_set = true;
442.                 }
443.                 if (board[act_pos.row][act_pos.col].is_open == 0)
444.                     OpenBlock(act_pos, game_state, pmap_info);
445.                 else
446.                     OpenSurround(act_pos, game_state, pmap_info);
447.             }else if (msg.is_right())//鼠标右键改变标记
448.                 ChangeState(&board[act_pos.row][act_pos.col], &(pmap_info->map_flag));
449.         }
450.     }
```

```
451.     return act_pos;
452. }
453.
454. //================================================
455. //函数名：DrawBoundary
456. //功能：绘制网格
457. //输入参数：指向地图信息结构体的结构体指针
458. //返回值：无
459. //================================================
460. void DrawBoundary(MAP_INFO* pmap_info)
461. {
462.     int map_row = pmap_info->map_row;
463.     int map_col = pmap_info->map_col;
464.     setcolor(EGERGB(128, 128, 128));
465.     setlinewidth(1);
466.     int point_pair1[16]={10, 10, 10, 10+2+18*map_row+2,
467.                          11, 10, 11, 10+2+18*map_row+1,
468.                          10, 10, 10+2+18*map_col+2, 10,
469.                          10, 11, 10+2+18*map_col+1, 11};
470.     drawlines(4, point_pair1);//画左上边界线
471.
472.     for (int i=0; i<=map_col; i++)
473.         line(12+i*18, 12, 12+i*18, 12+map_row*18);//画竖网格线
474.     for (int i=0; i<=map_row; i++)
475.         line(12, 12+i*18, 12+map_col*18, 12+i*18);//画横网格线
476.
477.     setcolor(EGERGB(255, 255, 255));
478.     int point_pair2[16]={11, 10+2+18*map_row+1, 12+map_col*18+2, 10+2+18*map_row+1,
479.                          10, 10+2+18*map_row+2, 12+map_col*18+2, 10+2+18*map_row+2,
480.                          12+map_col*18+1, 11, 12+map_col*18+1, 10+2+18*map_row+2,
481.                          12+map_col*18+2, 10, 12+map_col*18+2, 10+2+18*map_row+2};
482.     drawlines(4, point_pair2);//画右下边界线
483. }
484.
485. //================================================
486. //函数名：DrawImage
487. //功能：绘制网格图片
488. //输入参数：游戏图片数组(PIMAGE*)，指向地图信息结构体的结构体指针，游戏运行状态
489. //返回值：无
490. //================================================
491. void DrawImage(PIMAGE* game_img, MAP_INFO* pmap_info, int game_state)
492. {
493.     for (int i=1; i<=pmap_info->map_row; i++){
494.         for (int j=1; j<=pmap_info->map_col; j++){
495.             int x, y, img_type;
496.             PosToMouse(MAP_POS(i, j), &x, &y);
497.
498.             if (board[i][j].is_open == 0){//格子没有被打开时
499.
```

```
500.                 switch (board[i][j].state){
501.                 case 0://没有标记时
502.                     if (game_state == 0)//游戏还在进行时
503.                         img_type = board[i][j].is_down ? 0 : BLANK;//未打开的格子被按下时，展示为 0 雷格，若未按下，展示为未打开格
504.                     else if (game_state == 1)//游戏失败后
505.                         img_type = (board[i][j].num == 15) ? MINE : BLANK;//未打开的含雷格展示为雷格，若未含雷，展示为未打开格
506.                     else//游戏胜利后
507.                         img_type = BLANK;//未打开格均展示为未打开格
508.                     break;
509.                 case 1://标志为旗帜时
510.                     if (game_state == 1)//游戏失败后
511.                         img_type = (board[i][j].num != 15) ? MISTAKE : MINE;//标为旗帜的格子没有含雷，展示为错误格，若含雷，展示为雷格
512.                     else//游戏正常运行或胜利后
513.                         img_type = FLAG;//标为旗帜的格子展示为旗帜
514.                     break;
515.                 case 2://标记为问号后
516.                     img_type = (game_state == 1 && board[i][j].num == 15) ? MINE : QUESTION;// 游戏失败且含雷时，展示为雷格，否则展示为问号
517.                     break;
518.                 }
519.
520.             }else//格子被打开后
521.                 img_type = (board[i][j].num != 15) ? board[i][j].num : BLOOD;//若不是雷，展示为数字格，否则展示为红雷格
522.
523.             putimage(x, y, game_img[img_type]);
524.         }
525.     }
526. }
527.
528. //=====================================================
529. //函数名：DrawImage
530. //功能：绘制网格图片
531. //输入参数：游戏图片数组(PIMAGE*)，是否展示惊讶的布尔值(bool)，游戏运行状态，地图的列数
532. //返回值：无
533. //=====================================================
534. void DrawEmoji(PIMAGE* game_img, bool is_surprised, int game_state, int map_col)
535. {
536.     int emoji_type;
537.     switch (game_state){
538.     case 0://游戏正常进行
539.         emoji_type = is_surprised ? SURPRISE : SMILE;
540.         break;
541.     case 1://游戏失败
542.         emoji_type = DEAD;
543.         break;
```

```
    case 2://游戏胜利
        emoji_type = WIN;
        break;
    }
    putimage_withalpha(NULL, game_img[emoji_type], 10+2+18*map_col+3+140 - 100, 15);
}
```

load.h

```
#pragma once

#define NONE -1//定义按钮空ID

enum GAME_IMG{
    BLANK=9, FLAG, QUESTION, MINE, BLOOD, MISTAKE, SMILE, SURPRISE, DEAD, WIN
};
//枚举图片对应图片数组的下标值，便于使用

#include <graphics.h>
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include "calc.h"

//=================================================
//函数名：LoadButtonImage
//功能：创建并加载按钮会使用到的图片
//输入参数：存放按钮图片的数组(PIMAGE*)
//返回值：无
//=================================================
void LoadButtonImage(PIMAGE* img_array);

//=================================================
//函数名：LoadGameImage
//功能：创建并加载游戏中会使用到的图片
//输入参数：存放游戏图片的数组(PIMAGE*)
//返回值：无
//=================================================
void LoadGameImage(PIMAGE* img_array);

//=================================================
//函数名：ReleaseImage
//功能：释放图片数组的内存
//输入参数：需要释放的图片数组(PIMAGE*)，图片数量(int)
//返回值：无
//=================================================
void ReleaseImage(PIMAGE* img_array, int n);

//=================================================
//函数名：SaveGame
```

```
42. //功能：保存当前未完成的游戏
43. //输入参数：地图信息结构体的指针(MAP_INFO*)，地图本身(BLOCK(*)[32])
44. //返回值：0 为保存成功，1 为保存失败
45. //=================================================
46. int SaveGame(const MAP_INFO* map_info, const BLOCK board[][32]);
47.
48. //=================================================
49. //函数名：LoadSave
50. //功能：加载存档信息
51. //输入参数：地图信息结构体的指针(MAP_INFO*)，地图本身(BLOCK(*)[32])
52. //返回值：0 为加载成功，1 为加载失败
53. //=================================================
54. int LoadSave(MAP_INFO* map_info, BLOCK board[][32]);
55.
56. //=================================================
57. //函数名：RankUpdate
58. //功能：尝试更新排名信息
59. //输入参数：排名信息数组(RANK_INFO*)，地图信息结构体的指针(MAP_INFO*)，所用时间(int)
60. //返回值：无
61. //=================================================
62. void RankUpdate(RANK_INFO rank_info[3], const MAP_INFO* pmap_info, int time);
63.
64. //=================================================
65. //函数名：LoadRank
66. //功能：加载排名信息
67. //输入参数：排名信息数组(RANK_INFO*)
68. //返回值：无
69. //=================================================
70. void LoadRank(RANK_INFO* rank_info);
71.
72. //=================================================
73. //函数名：SaveRank
74. //功能：保存排名信息
75. //输入参数：排名信息数组(RANK_INFO*)
76. //返回值：无
77. //=================================================
78. void SaveRank(RANK_INFO* rank_info);
```

load.cpp

```
1.  #include "load.h"
2.
3.  //=================================================
4.  //函数名：LoadButtonImage
5.  //功能：创建并加载按钮会使用到的图片
6.  //输入参数：存放按钮图片的数组(PIMAGE*)
7.  //返回值：无
8.  //=================================================
9.  void LoadButtonImage(PIMAGE* img_array)
10. {
```

```
11.     char name[2][16] = {"button_pop", "button_click"};
12.     for (int i=0; i<2; i++){
13.         char buf[64];
14.         sprintf(buf, "resource\\%s.png", name[i]);
15.         img_array[i] = newimage();
16.         if (grOk != getimage(img_array[i], buf)){
17.             MessageBoxA(getHWnd(), "图片资源加载失败！", "错误", MB_ICONWARNING | MB_APPLMODAL);
18.             exit(0);
19.         }
20.     }
21. }
22.
23. //=================================================
24. //函数名：LoadGameImage
25. //功能：创建并加载游戏中会使用到的图片
26. //输入参数：存放按钮图片的数组(PIMAGE*)
27. //返回值：无
28. //=================================================
29. void LoadGameImage(PIMAGE* img_array)
30. {
31.     for (int i=0; i<19; i++)
32.         img_array[i] = newimage();
33.
34.     for (int i=0; i<=8; i++){
35.         char str[40];
36.         sprintf(str, "resource\\%d.gif", i);
37.         if (grOk != getimage(img_array[i], str)){
38.             MessageBoxA(getHWnd(), "图片资源加载失败！", "错误", MB_ICONWARNING | MB_APPLMODAL);
39.             exit(0);
40.         }
41.     }
42.     char name[10][16] = {"blank.gif", "flag.gif", "question.gif", "mine.gif", "blood.gif", "mista
    ke.gif", "smile.png", "surprise.png", "dead.png", "win.png"};
43.     for (int i=9; i<=18; i++){
44.         char buf[64];
45.         sprintf(buf, "resource\\%s", name[i-9]);
46.         if (grOk != getimage(img_array[i], buf)){
47.             MessageBoxA(getHWnd(), "图片资源加载失败！", "错误", MB_ICONWARNING | MB_APPLMODAL);
48.             exit(0);
49.         }
50.     }
51. }
52.
53. //=================================================
54. //函数名：ReleaseImage
55. //功能：释放图片数组的内存
56. //输入参数：需要释放的图片数组(PIMAGE*)，图片数量(int)
57. //返回值：无
58. //=================================================
```

```
59. void ReleaseImage(PIMAGE* img_array, int n)
60. {
61.     for (int i=0; i<n; i++)
62.         delimage(img_array[i]);
63. }
64.
65. //================================================
66. //函数名：SaveGame
67. //功能：保存当前未完成的游戏
68. //输入参数：地图信息结构体的指针(MAP_INFO*)，地图本身(BLOCK(*)[32])
69. //返回值：0 为保存成功，1 为保存失败
70. //================================================
71. int SaveGame(const MAP_INFO* pmap_info, const BLOCK board[][32])
72. {
73.     srand(time(0));
74.     FILE* fp = fopen("data\\save.txt", "w");
75.     if (fp == NULL){
76.         fclose(fp);
77.         MessageBoxA(getHWnd(), "存档保存失败", "警告", MB_ICONERROR | MB_APPLMODAL);
78.         return 1;
79.     }else{
80.         fprintf(fp, "%d\n", EncryptNum(pmap_info->map_row));
81.         fprintf(fp, "%d\n", EncryptNum(pmap_info->map_col));
82.         fprintf(fp, "%d\n", EncryptNum(pmap_info->map_mines));
83.         fprintf(fp, "%d\n", EncryptNum(pmap_info->map_flag));
84.         fprintf(fp, "%d\n", EncryptNum(pmap_info->map_open));
85.         fprintf(fp, "%d\n", EncryptNum(pmap_info->run_time));
86.         fprintf(fp, "%d\n", EncryptNum(pmap_info->map_row) + EncryptNum(pmap_info->map_col) + Enc
    ryptNum(pmap_info->map_mines)\
87.                             + EncryptNum(pmap_info->map_flag) + EncryptNum(pmap_info->map_open) + E
    ncryptNum(pmap_info->run_time));
88.         for (int i=1; i<=pmap_info->map_row; i++){
89.             for (int j=1; j<=pmap_info->map_col; j++){
90.                 fprintf(fp, "%d\n", EncryptBlock(board[i][j]));
91.             }
92.         }
93.         fclose(fp);
94.         MessageBoxA(getHWnd(), "存档保存成功", "成功", MB_ICONINFORMATION | MB_APPLMODAL);
95.         return 0;
96.     }
97. }
98.
99. //================================================
100. //函数名：LoadSave
101. //功能：加载存档信息
102. //输入参数：地图信息结构体的指针(MAP_INFO*)，地图本身(BLOCK(*)[32])
103. //返回值：0 为加载成功，1 为加载失败
104. //================================================
105. int LoadSave(MAP_INFO* pmap_info, BLOCK board[][32])
```

```
106. {
107.     FILE* fp = fopen("data\\save.txt", "r");
108.     if (fp == NULL){
109.         fclose(fp);
110.         MessageBoxA(getHWnd(), "存档打开失败", "警告", MB_ICONERROR | MB_APPLMODAL);
111.         return 1;
112.     }else{
113.         int save_mines=0, save_flag=0, save_open=0;//记录存档实际含有的雷数，旗帜数与打开的格子数
114.         bool save_damaged = false;
115.
116.         int a, b, c, d, e, f, g;
117.         if (7 != fscanf(fp, "%d%d%d%d%d%d%d", &a, &b, &c, &d, &e, &f, &g) || a+b+c+d+e+f!=g)// 读
    取地图基本信息
118.             save_damaged = true;
119.         else{
120.             pmap_info->map_row = DecryptNum(a);
121.             pmap_info->map_col = DecryptNum(b);
122.             pmap_info->map_mines = DecryptNum(c);
123.             pmap_info->map_flag = DecryptNum(d);
124.             pmap_info->map_open = DecryptNum(e);
125.             pmap_info->run_time = DecryptNum(f);
126.         }
127.
128.         for (int i=1; i<=pmap_info->map_row && !save_damaged; i++){
129.             for (int j=1; j<=pmap_info->map_col && !save_damaged; j++){
130.
131.                 int key;
132.                 if(1 != fscanf(fp, "%d", &key))//读取地图每格信息
133.                     save_damaged = true;
134.                 board[i][j] = DecryptBlock(key);
135.
136.                 if (board[i][j].num == 15)
137.                     save_mines++;
138.                 if (board[i][j].is_open == 1)
139.                     save_open++;
140.                 if (board[i][j].state == 1)
141.                     save_flag++;
142.             }
143.         }
144.
145.         if (save_mines!=pmap_info->map_mines || save_open!=pmap_info->map_open || save_flag!=pma
    p_info->map_flag)
146.             save_damaged = true;
147.         fclose(fp);
148.
149.         if (save_damaged){
150.             MessageBoxA(getHWnd(), "存档文件损坏，无法打开", "警告", MB_ICONERROR | MB_APPLMODAL);
151.             return 1;
152.         }else
```

```
153.             return 0;
154.     }
155. }
156.
157.
158. //=================================================
159. //函数名：RankUpdate
160. //功能：尝试更新排名信息
161. //输入参数：排名信息数组(RANK_INFO*)，地图信息结构体的指针(MAP_INFO*)，所用时间(int)
162. //返回值：无
163. //=================================================
164. void RankUpdate(RANK_INFO rank_info[3], const MAP_INFO* pmap_info, int time)
165. {
166.     int difficulty;
167.     if (pmap_info->map_row==9 && pmap_info->map_col==9 && pmap_info->map_mines==10)
168.         difficulty=0;
169.     else if (pmap_info->map_row==16 && pmap_info->map_col==16 && pmap_info->map_mines==40)
170.         difficulty=1;
171.     else if (pmap_info->map_row==16 && pmap_info->map_col==30 && pmap_info->map_mines==99)
172.         difficulty=2;
173.     else
174.         return;
175.
176.     if (time <= rank_info[difficulty].time){
177.         char username[33];
178.         DWORD dwSize_username = 33;
179.         GetUserNameA(username, &dwSize_username);
180.         rank_info[difficulty].time = time;
181.         strcpy(rank_info[difficulty].username, username);
182.         SaveRank(rank_info);
183.     }
184. }
185.
186.
187. //=================================================
188. //函数名：LoadRank
189. //功能：加载排名信息
190. //输入参数：排名信息数组(RANK_INFO*)
191. //返回值：无
192. //=================================================
193. void LoadRank(RANK_INFO* rank_info)
194. {
195.     FILE* fp = fopen("data\\rank.txt", "r");
196.     if (fp == NULL){
197.         for (int i=0; i<3; i++){
198.             strcpy(rank_info[i].username, "-");
199.             rank_info[i].time = 99999;
200.         }
201.     }else{
```

```
202.         for (int i=0; i<3; i++){
203.             fscanf(fp, "%s", rank_info[i].username);
204.             fscanf(fp, "%d", &rank_info[i].time);
205.         }
206.     }
207.     fclose(fp);
208. }
209.
210. //=================================================
211. //函数名：SaveRank
212. //功能：保存排名信息
213. //输入参数：排名信息数组(RANK_INFO*)
214. //返回值：无
215. //=================================================
216. void SaveRank(RANK_INFO* rank_info)
217. {
218.     FILE* fp = fopen("data\\rank.txt", "w");
219.     if (fp == NULL)
220.         MessageBoxA(getHWnd(), "排名保存失败", "警告", MB_ICONERROR | MB_APPLMODAL);
221.     else
222.         for (int i=0; i<3; i++){
223.             fprintf(fp, "%s\n", rank_info[i].username);
224.             fprintf(fp, "%d\n", rank_info[i].time);
225.         }
226.     fclose(fp);
227. }
```

分析总结、收获和体会:

优点：
注释充足，充分的应用了所学知识，并在其基础上学习了 ege 及 C++有关内容，并正确使用。
成品的可玩性较用控制台操作的版本大大提高。

创新之处：
菜单的表情包；
使用 emoji 替代了传统的小黄脸；
对存档有加密。

不足之处：
本想使用动态二维结构体数组作为地图的存储方式，但碍于麻烦，以及由于游戏本身对该项要求的不必要性，没有对其进行实现。
太长了，别人不爱看。

需要改进的地方：
可使用 C++的语法及函数来精简程序
可添加自动标雷的功能

收获与学习体会：
通过这次大作业，本人收获了很多，包括但不限于：深刻理解了扫雷游戏的开发，能使用 ege 来制作自己想要的图

形界面，了解了部分 C++语法，提高了自己的编程能力，养成了写注释的好习惯。		
自我评价：	是	否
程序运行是否无 bug？		√
是否在撰写报告之前观看了 MOOC 里的华为编程实践和代码规范相关的视频？	√	
程序代码是否符合华为代码规范(对齐与缩进，变量名和函数名命名，必要的注释等)？		√
是否按模块化要求进行了程序设计，系统功能是否完善？	√	
是否是独立完成？	√	

自我评语：

在我本人看来，本次的大作业完成度很高，达到了我本人的预期，我对这次大作业的完成十分满意。

报告完成日期：2022/12/5