踩地雷遊戲

實習目標：

丁培毅

1. 實作各種函式
2. 二維陣列記錄地雷區以及遊戲狀態
3. 運用 enum 自訂資料型態
4. 練習迴圈控制
5. 輸入驗證迴圈
6. 簡易文字模式界面函式 utilwin32.c, utilwin32.h
   clrscr(), gotoxy(), setTextColor()
7. Non-blocking 輸入: kbhit() 與 getch()
8. Win32 圖形化介面與事件驅動程式設計

踩地雷遊戲

1. 程式範例輸出如右：
2. 使用者可以輸入列數, 行數, 以及地雷個數
3. 使用者每次可以輸入 **0 i j** 決定要打開座標點**(i,j)**, 如果該座標有地雷, 使用者就輸了 (此時程式會把未標示但是有地雷的點打開), 如果該座標沒有地雷, 程式會印出該座標點四周八格有幾個地雷, 如果該座標點四週八格沒有地雷, 程式會把四週八格一一打開, 並且進一步印出該座標點四周八格有幾個地雷
4. 如果使用者判斷該點有地雷, 可以輸入 **1 i j** 決定標示座標點**(i,j)**, 每增加標示一個地雷 (不論是對的還是錯的), 程式會減少上方剩餘未標示地雷數目
5. 如果使用者懷疑先前的標示有錯, 可以再次輸入 **1 i j** 來改變標示爲『?』
6. 如果目前標示『?』, 再次輸入 **1 i j** 可以還原成未標示
7. 如果使用者把所有沒有地雷的地方都開啓了, 使用者就贏了這一盤

```
請輸入列數(<=16), 行數(<=36),
      及地雷個數(>0) :10 10 10
還有 10 個地雷未標示
    0 1 2 3 4 5 6 7 8 9
  0 - - - - - - - - - -
  1 - - - - - - - - - -
  2 - - - - - - - - - -
  3 - - - - - - - - - -
  4 - - - - - - - - - -
  5 - - - - - - - - - -
  6 - - - - - - - - - -
  7 - - - - - - - - - -
  8 - - - - - - - - - -
  9 - - - - - - - - - -
請輸入 0:打開/1:標示 及 座標(i,j) ? 0 8 8
```

```
還有 10 個地雷未標示
    0 1 2 3 4 5 6 7 8 9
  0 - - - - - - - - - -
  1 - - - - - - - - - -
  2 - - - - - - - - - -
  3 - - - - - - - - - -
  4 - - - - - - - - - -
  5 - - - - - - 1 1 2 -
  6 - - - - 3 1 1 0 1 1
  7 - - - - 1 0 0 0 0 0
  8 1 1 1 1 1 0 0 0 0 0
  9 0 0 0 0 0 0 0 0 0 0
請輸入 0:打開/1:標示 及 座標(i,j) ?
```

界面修改版

還有 10 個地雷未標示

	0	1	2	3	4	5	6	7	8	9
0	■	■	■	■	■	■	■	■	■	■
1	■	■	■	■	■	■	■	■	■	■
2	1	2	■	■	■	■	■	■	■	■
3	0	1	■	■	■	■	■	■	■	■
4	0	1	■	■	■	■	■	■	■	■
5	0	1	1	1	1	■	■	2	■	■
6	0	0	0	0	1	■	■	■	■	■
7	0	0	0	0	1	2	3	■	■	■
8	0	0	0	0	0	0	1	■	■	■

請輸入 打開:0/標示:1 及 座標(i,j) ?

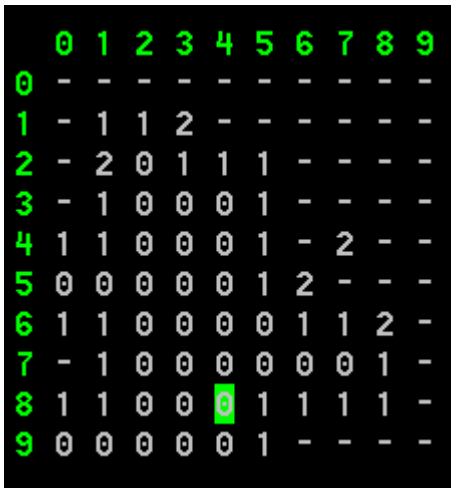
也可以把未知格, 標示格以及遊戲盤面換成如下 Big5 字元

	0	1	2	3	4	5	6	7	8	9
0	1	2	◤	1	0	0	1	◤	1	0
1	◤	2	1	2	1	1	1	1	1	0
2	1	1	0	1	◤	1	0	0	0	0
3	1	1	1	1	2	2	1	0	0	0
4	1	◤	1	0	1	◤	1	0	0	0
5	1	1	1	0	1	1	1	0	0	0
6	1	1	0	1	2	2	1	0	0	0
7	◤	2	1	2	◤	◤	1	0	0	0
8	1	2	◤	2	2	2	1	0	0	0

恭喜!! 你贏了這盤
再來一盤 (y/n) ? n

鍵盤游標界面版

還有 10 個地雷未標示　請用 空白 鍵翻開,
按 **m** 標示/改變標示, 以 ↑ ↓ ← → 移動

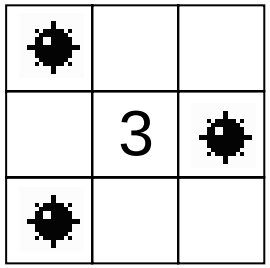


1. 程式範例輸出如右：
2. 本遊戲中使用者可以輸入遊戲的列數, 行數, 以及地雷個數
3. 使用者可以用 ↑ ↓ ← → 按鍵移動, 按 空格 鍵決定要打開游標位置的座標點, 如果該座標有地雷, 使用者就輸了 (此時程式會把未標示但是有地雷的點翻開)
   如果該座標沒有地雷, 程式會印出該座標點四周八格有幾個地雷, 如果該座標點四週八格沒有地雷, 程式會把四週八格一一打開, 並且進一步印出該座標點四周八格有幾個地雷
4. 如果使用者判斷該點有地雷, 可以按 **m** 決定標示目前游標位置的座標點, 每多標示一個地雷 (不論是對的還是錯的), 程式會減少上方剩餘未標示地雷數目
5. 如果使用者懷疑先前的標示有錯, 可以再次按 **m** 來改變標示爲『?』
6. 如果目前標示『?』, 再次按 **m** 可以還原成未標示
7. 如果使用者把所有沒有地雷的地方都開啓了, 使用者就贏了這一盤

分析

1. 這個題目需要使用 stdlib.h 裡提供的虛擬亂數產生器 rand() 以及初始化函式 srand()，一般常用 time.h 裡的 time() 函式來取得 1970/01/01 00:00:00 到現在之間的秒數來作爲亂數種子，如果需要產生均勻分佈在 0~(n-1) 之間的整數，將這個均勻分佈在 [0,1) 的浮點數乘 n 再轉爲整數 (int)((double) rand() / RAND_MAX * n) 即可，平常偷懶一點的方法是 rand()%n，均勻分佈的特性較差。在這個程式裡，要使用亂數來挑選地雷的位置，二維的場地可以直接挑兩個亂數 (列, 行)，也可以看成是一維的 (一列一列接續起來)，如果有 n 格，在這裡面要挑選 m 個位置來放地雷，如果要求非常均勻，可以用撲克牌發 5 張牌判斷牌型那裡有兩種方法可以做，如果比較不在意是不是那麼均勻，可以連續挑 0~(n-1) 範圍裡的亂數，挑出來以後再檢查是不是已經挑過了。請記得用函式撰寫以下各步驟。
2. 上面在記錄的時候需要設計二維的陣列 int mines[SIZE][SIZE], 這個陣列裡面的元素先初始化爲 0, 挑到某個位置要放地雷時，設爲 -1

2. 因爲在遊戲過程中需要印出來每一格四周九宮格裡有幾個地雷，在地雷位置放好以後這些資料就是固定的了，放置地雷的地方是不需要算的，因爲如果不小心打開地雷那一格，當場就爆了，程式不需要印出四周有幾個地雷，如右圖，對每一格 (row, col) 來說，需要寫類似下圖的兩層的迴圈來加總




```
for (i=-1, mines[row][col]=0; i<=1; i++)
    for (j=-1; j<=1; j++) {
        if ((i==0)&&(j==0) continue;
        if (mines[row+i][col+j]==-1)
            …
    }
```

3. 步驟 2 的陣列 int mines[SIZE][SIZE] 可以記住地雷的位置，就在 mines[row][col] 的地方紀錄 -1，當然還有另外一種紀錄方法，就是用 int minesCoordinates[20][2]; 這裡假設最多20 個地雷，然後每個地雷有 (row, col) 兩個欄位，這種表示法當地雷數目不會太多時比上面的 mines 陣列省空間，而且這種紀錄方法很容易找到所有地雷的位置，不過測試某一個座標點有沒有地雷時比較慢，需要用迴圈去一個一個比對，因爲比較浪費空間，所以我們就把第二種不同的資訊 – 該格點四周地雷數目 – 都一起紀錄在這個陣列裡。

4. 玩這個遊戲的時候，使用者基本上輸入一個打開的動作和一個座標，然後程式檢查有沒有誤觸地雷，顯示那一格四周有幾個地雷，然後不斷重複這兩個動作 ⇒ 程式的主體應該就如右圖是一個簡單的迴圈

```
while (遊戲尚未結束) {
    ❶ 讀取使用者指定的動作和座標點
    ❷ 顯示已經被打開的那些格子的四周地雷數
}
```

5. 觀察範例執行程式的顯示，你可以看到除了顯示那些被打開的格子外，其它格子、座標軸、格線，也都需要顯示出來，而且在文字模式的輸出時，其實列印一整列是比較簡單的，要一個格子一個格子獨自輸出反而比較麻煩，所以上面的 ❷ 可以改成顯示整個畫面，反正包涵了剛打開的 (其實更進一步的視窗圖形介面也都是用這樣的想法做出來的，這樣子可以把所有的繪圖程式碼都集中在一起，程式中必須考量繪圖這個動作不是只有程式運作邏輯自己決定要不要做的，操作應用程式的人也有這個權力: 可以把視窗從別的視窗後面拉出來，可以縮小及還原視窗，這些動作都無關程式邏輯)

```
while (遊戲尚未結束) {
    ❶ 讀取使用者指定的動作和座標點
    ❷ 顯示整個畫面
}
```

6. 再進一步想一下，光光有 mines 陣列還沒有辦法畫出程式要的畫面，比如說某一個元素 mines[1][2] 是 -1，代表這裡有一個地雷，可是我們不見得要畫出地雷啊? 有的時候畫未打開的狀態，有的時候畫已標示的狀態，只有當玩家踩到這個地雷時才畫地雷，所以其實每一格還需要有一個狀態變數記錄目前這格到底翻開了沒，標示了沒…，所以也是一個二維陣列 status
7. 可以練習 enum 的語法，把每一格可能的狀態定義出來，例如：未翻開、已翻開、已標示、標示爲『?』，標準範例如下：

```
enum Color {red, green, blue};        enum Color array[20][30];
enum x;   x = red;
```

8. 目前的繪圖函式其實就根據 mines 和 status 兩個二維陣列，寫兩層迴圈把每一格的資料先列印出來，在更進一步修改繪圖介面之前，我們先完成玩家的兩個動作：標示和翻開 (請寫兩個函式來完成)

   標示：基本上就是要改變 status[row][col] 的狀態，由 未開啓變成已標示，或是由已標示變成未確定，或是由未確定變回未開啓；同時也需要考量玩家的操作錯誤，例如標示一個已開啓的格子，這時應該不要有任何反應；另外因爲在遊戲畫面左上角顯示剩餘幾個地雷沒有標示，所以程式應該要再準備一個變數，一開始設爲總地雷數目，每標示一個就減一，雖然有可能會標示錯誤，但是還是一個進度的參考數值

   翻開：當玩家決定翻開某一格時，第一是要檢查這裡有沒有地雷，如果有的話遊戲就結束了，遊戲這樣子結束的話，應該要在畫面上顯示清楚是在哪一格踩到地雷的，在範例執行程式裡也順便把所有的地雷一起翻開來讓玩家可以檢視；第二是看看翻開的這格四周有幾個地雷，除了把這格的狀態設成已開啓之外，如果四周是沒有地雷的，其實就可以直接把四周一一翻開，這個動作其實就和翻開自己是一樣的，所以這裡應該把這個翻開的函式設計成遞迴函式，特別注意結束條件

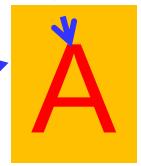
簡易文字模式繪圖工具

9. 在 utilwin32.c 裡面提供了幾個簡單的繪圖工具函式，但是在寫這種文字模式遊戲時蠻有用的，包括

clrscr(): 清除螢幕內容

gotoxy(int irow, int icol): 移動游標到座標點 (irow, icol) 的地方，其中 irow 的範圍是 1~24, icol 的範圍是 1~80

WORD setTextColor(WORD color): 設定接下來在螢幕上輸出文字的顏色，顏色由下列常數組合出來，其中文字前景顏色由
FOREGROUND_RED, FOREGROUND_GREEN,
FOREGROUND_BLUE, FOREGROUND_INTENSITY
以位元的 OR 組合出來，文字背景顏色由
BACKGROUND_RED, BACKGROUND_GREEN,
BACKGROUND_BLUE, BACKGROUND_INTENSITY
以位元的 OR 組合出來, 回傳修改前的顏色, 可以用來回復顏色

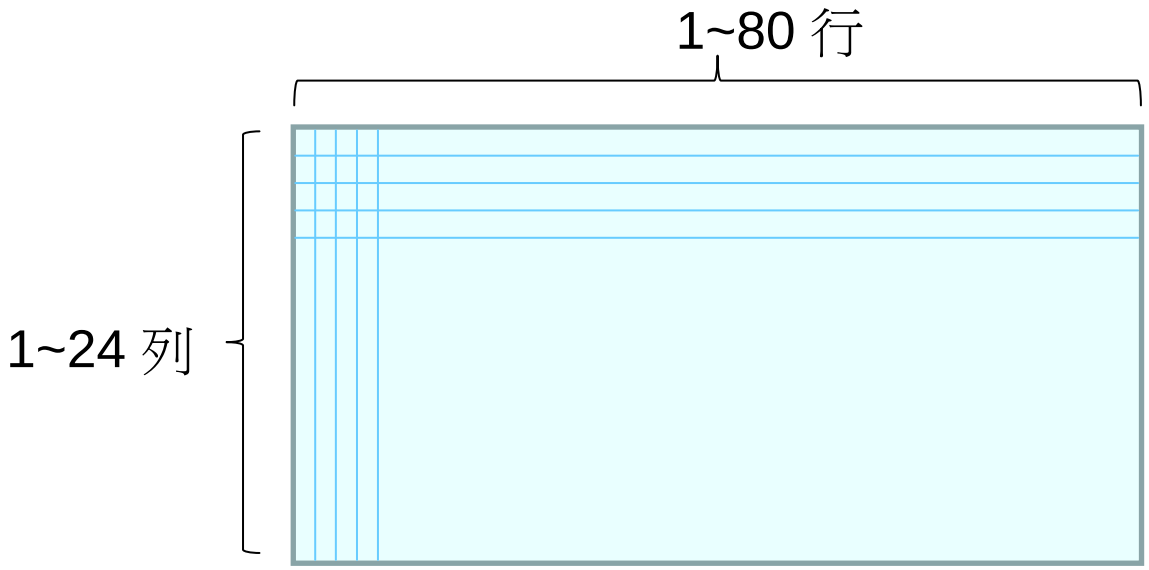


任何檔案中的程式如果需要使用上述函式就需要以下列敘述

```
#include “utilwin32.h”
```

在程式中引入 .h 定義檔一起編譯, VC 命令列如下

```
cl mimesweeper.c utilwin32.c
```

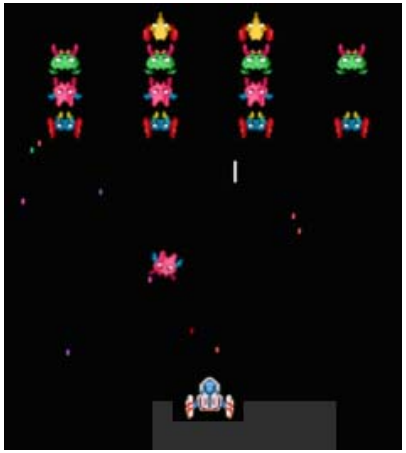


10. 如上圖文字模式視窗裡一般可以顯示 24x80 個字元 (可以手動設定其它大小)，stdio 函式庫運作時有一個目前游標的位置，列印時由游標位置一個字元一個字元輸出，游標位置就移動到最後一個輸出字元的下一個位置，stdio 函式庫並沒有提供操作游標位置的工具函式，一切輸出都按照順序執行。像前面所談到在這個遊戲的界面裡，有一個迴圈不斷地詢問使用者輸入，然後印出整個遊戲畫面，螢幕會不斷地向上捲動，對於玩家來說會造成很大的干擾，沒有辦法輕易地掌握盤面的變化。所以基本上希望輸出的界面裡所有的元素位置都是固定的，要達到這個效果，就需要 **gotoxy(行, 列)** 這個移動游標位置的函式。呼叫函式 gotoxy(x, y) 會把游標移動到第 x 列，第 y 行的位置，接下來的 printf, putchar.. 等等輸出就會由該點繼續下去

11. 在前兩個版本裡，玩家需要輸入命令 (0:翻開/1:標示) 以及格點座標 (row,col)，這使得玩家需要操作至少 10 個數字按鍵，其實是很困擾的，在第三個版本裡，我們修改一下界面，讓玩家只需要運用 上下左右 按鍵來移動想要選取的格點，另外用 空白鍵 以及按鍵 **m** 來決定翻開與標示 的動作，這樣的改變雖然不是很多，但是在撰寫程式的觀念需要稍微修改 – 本來程序化的程式要求使用者輸入的時機是完全由程式來控制的，程式執行到 scanf() 時，使用者才能配合著透過鍵盤輸入資料，使用者如果不配合的話，程式也只能乖乖停在那裡等候，程式在執行其它程序的時候，使用者完全無法輸入資料，在第三個版本裡，使用者任何時候按下控制鍵時，程式似乎都會配合處理，也就是程式需要無時無刻都在等候輸入，才能夠即時處理玩家的控制按鍵。同時 scanf() 或是 getchar() 輸入時都要多按 一個 <enter> 才讀得到，這也是很討厭的，程式的最後架構像是右側的迴圈，這樣就可以達到上面的目標了。

```
while (1) {
    read user input with getch()
    process user input
    change the output in response
}
```

12. 這個架構會有一個問題，就是使用者不輸入時，程式就停在讀取輸入的地方，在掃地雷這個程式裡還不是太大問題，因爲使用者不輸入的時候其實電腦也不需要做其他事，但是你可以想像一些其它的遊戲，



12. 玩家沒有按任何按鍵或是輸入資料時，程式需要做很多顯示的動作，例如上面三個遊戲: 小密蜂、小精靈PACMAN、超級馬利歐都是這樣的，遊戲世界中所有角色的狀態一直在改變，所以顯示在界面的狀態也會不斷地改變，如果像是剛才講的架構 - 在使用者不輸入時程式完全暫停下來等候，就沒有需要的效果了，所以這裡再介紹一個簡單的函式可以不要改變程式環境太多就得到所需要的效果，在 conio 中有一個 kbhit() 函式可以測試鍵盤是否有輸入按鍵，這是一個所謂 non-blocking 的函式，當使用者有按鍵時，kbhit() 回傳 1，程式可以再透過 getch() 讀取，如果沒有按鍵時 kbhit() 會傳回 0，程式可以做其它的背景動作，因爲 CPU 的速度很快，可以用 delay 調節角色的速度

```
while (1) {
    if (kbhit() && getch()) process user input
    change the output according to the state
    delay(a short period)
}
```

13. 接下來說明一下鍵盤上功能鍵的處理：

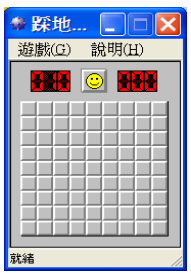
首先，stdio 的 scanf() 或是 getchar() 是完全不處理 F1~F12 以及 ↑ ↓ ← → Home End PgUp PgDown 這些功能鍵的，需要運用 conio 裡面的 getch() 或是 getche()，使用者按下可顯示的按鍵時 (a~z, 0~9, !@#$...)，getch() 傳回按鍵的 ASCII 數值，如果使用者按下功能鍵時，getch() 傳回 0xe0 代表是功能鍵，此時程式需要呼叫第二次 getch() 如果讀到 0x48 代表 ↑，0x50 代表 ↓，0x4b 代表 ←，0x4d 代表 →，其它特殊按鍵如右表，你可以用下列簡單程式測試一下

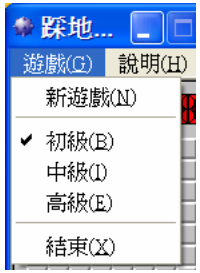
```
#include <conio.h>
#include <stdio.h>
void main() {
    int c, d;
    if ((c=getch()) == 0xe0 || c == 0) {
        d = getch();
        printf("%2x %2x\n", c, d);
    }
}
```

↑	0xe0	0x48
↓	0xe0	0x50
←	0xe0	0x4b
→	0xe0	0x4d
Insert	0xe0	0x52
Delete	0xe0	0x53
Home	0xe0	0x47
End	0xe0	0x4f
PgUp	0xe0	0x49
PgDn	0xe0	0x51
F1	0x00	0x3b
F2	0x00	0x3c
F4	0x00	0x3e
F5	0x00	0x3f
F6	0x00	0x40
F8	0x00	0x42
F9	0x00	0x43
F10	0x00	0x44
F11	0xe0	0x85
F12	0xe0	0x86

視窗版踩地雷程式

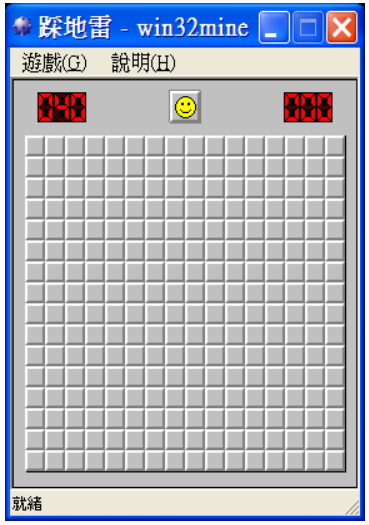
1. 前面寫了三個版本的掃地雷程式，但是因爲介面只有**鍵盤**和只**能顯示文字的視窗**，雖然已經意思到了，可是應用程式看起來實在不吸引人，在 Windows XP 裡面有附這個踩地雷的應用程式，接下來請嘗試透過下面提供的工具函式來做出一個視窗版的掃地雷程式

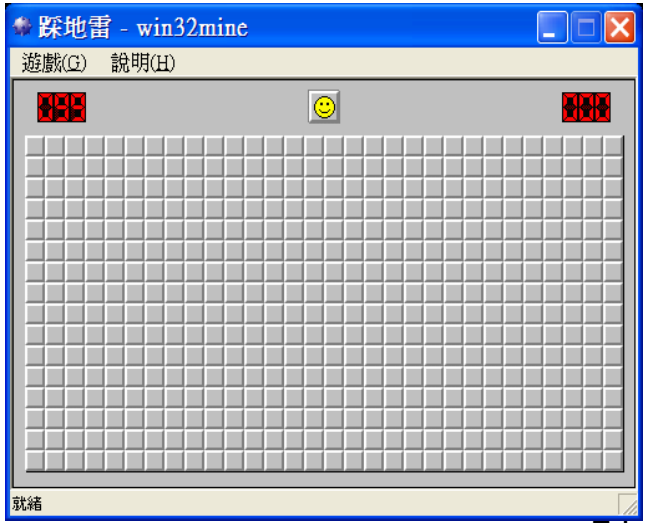




2. 請下載 **minesweeper UI framework** (971013win32mine_C_FacadeOnly.rar)

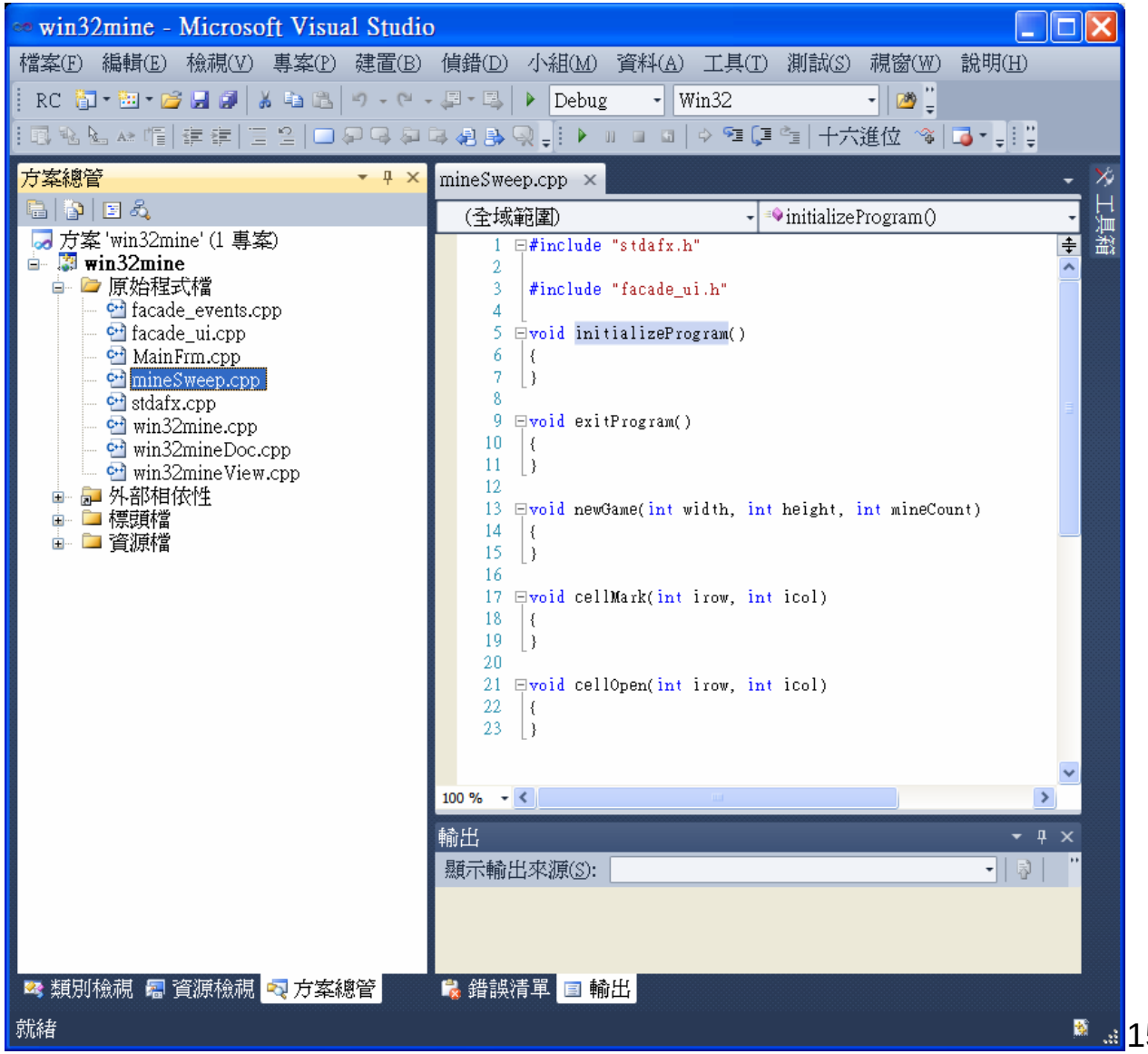
3. 解壓縮後在 win32mine 資料匣裡開啓 win32mine.sln，以 **vc2010** 建置並且執行，你會看到一個看起來有模有樣的介面，不過因爲還沒有把踩地雷的邏輯加進去，所以滑鼠怎樣操作，除了選 初級/中級/高級有反應之外，其它按鍵或選單好像都沒反應 (請下載 dbwin32.exe 並執行，再操作這個程式就會看到一些訊息反應了)，程式框架提供基本的界面如圖所示。左上角數字爲未標示的地雷個數，右上角數字爲經過的時間，笑臉是重新開始遊戲的按鈕 (和『遊戲/新遊戲』選單相同)





4. 在遊戲選單中設計三個等級，初級預設爲 9x9, 10 個地雷，中級爲 16x16, 40 個地雷，高級爲 30x16, 99 個地雷

5. 右圖爲 VC2010 的界面，請點選**方案總管**，點開**原始程式**檔，點選 mineSweep.cpp 這裡有幾個空的函式，以下我們一一解釋你需要做的事情以及界面程式已經做好的事情，希望你能夠由前面的文字界面程式裡很快地改出這個視窗版的程式



6. void initializeProgram();
   GUI 界面程式在程式一開始執行的時候會呼叫這個函式一次，所以如果你的程式有需要開啓檔案或是配置資源，可以在這裡做
7. void exitProgram();
   GUI 界面程式在玩家選擇 遊戲/結束 或是按下標題列的 x 時關掉視窗時會呼叫這個函式一次，如果有需要儲存狀態可以在這裡做
8. void newGame(int width, int height, int mineCount);
   GUI 界面程式在玩家每次按下『笑臉按鈕』或是選擇『遊戲/新遊戲』，『遊戲/初級』，『遊戲/中級』，『遊戲/高級』選單時都會呼叫這個函式一次，函式的參數是: 總列數, 總行數, 以及地雷數，你的程式應該用這些參數來初始化遊戲。目前的界面沒有設計一個讓玩家自己指定總列數，總行數，和地雷個數的對話盒，所以三級的困難度都是預先設定好的，將來如果擴充的話就不見得是固定的了
9. void cellMark(int irow, int icol);
   GUI 界面程式在玩家每次以滑鼠右鍵點選某一格時，會呼叫這個函式，傳入那一格的列數和行數，其中 irow 的範圍是 0~width-1， icol 的範圍是 0~height-1
10. void cellOpen(int irow, int icol);
    GUI 界面程式在玩家每次以滑鼠左鍵點選某一格時，會呼叫這個函式，傳入那一格的列數和行數

11. 上面幾個函式是你需要填入內容的，你應該已經發現沒有 main() 函式了，這個程式執行時的進入點在有圖形界面的 Win32 視窗程式裡不叫做 main() 了，而是 WinMain() 函式，可是也不需要你來寫，這個部份在 GUI 界面程式裡已經寫好了，你可能會發現去提供給你的原始程式碼裡面搜尋都找不到這個函式，這個故事比較長我們暫且跳過，不過重點是整個框架根本沒有打算讓你更改這個函式，這和這學期所有寫過的 C 程式就有一點差別了，有一點基本概念的差異，原因是視窗應用程式的模型不是程序式的，而是事件驅動 (event-driven) 的，或者說是基於物件 (object-based) 的，這樣子使用者才能夠順利的和所有的應用程式界接，使用者才能夠掌控電腦的表現，而不是由某一個程式掌控一切。你的應用程式現在要自己看成是一個功能獨立的物件去配合 GUI 框架運作，有自己的邏輯 (實作在前面的幾個函式裡)，有自己的資料儲存空間；平常比較適合的語言是物件導向的 C++，可以用類別的基本封裝來實作物件，以 C 程式來說，你可以用檔案來封裝這個物件，在 mineSweep.cpp 檔案中你可以用 static 的方式定義(全域)變數，如此別的檔案裡的模組不會誤用到這些變數，這些檔案層次的全域變數就可以作爲你這組函式之間交換資料和記錄狀態使用，你不需要去掌控 main() 函式，去設計裡面的資料了…

12. 接下來我們要讓你可以開始加入程式碼並且像先前所有要求的程式一樣，一步一步測試你的程式：首先請下載 dbwin32.exe 這個工具程式，並且把它執行起來
13. 在 Win32 視窗程式中，stdio 提供的文字界面函式幾乎都沒有作用了 (sprintf(), sscanf() 還是很好用的函式)，想要輸出一些訊息比較簡單的方式就是用 TRACE("show me the variable %d %s\n", ivar, strvar); 這個敘述執行到時資料會顯示在 dbwin32 視窗中，TRACE 的功能沒有 printf 強大，如果你遇見它不接受的格式命令時，請運用 sprintf 先轉成字串，再運用 TRACE 輸出
14. 請在前面的五個函式中分別加入 TRACE("initializeProgram\n"); … 編譯，執行，然後用滑鼠操作一下界面，觀察在 dbwin32 視窗中什麼時候會出現什麼訊息
15. 接下來再來想像一下踩地雷程式的運作與邏輯
    i. 會有兩個二維陣列 mines 和一個 status，兩個陣列都需要在許多函式中使用，所以應該定義在全域的地方爲 static 變數，需要在 newGame() 函式裡面配置(如果是動態的)以及初始化
    ii. cellMark() 和 cellOpen() 兩個函式就是檢視 mines 陣列，修改 status 陣列
    iii. exitProgram() 裡需要釋放記憶體(如果是動態的)

15. iv. 在前面的文字界面版本裡面都有如右圖的主要控制迴圈，控制迴圈結束時程式就結束了。在這個視窗版本的程式裡，當所有的地雷都標示出來或是踩到某一個地雷時，代表遊戲結束了，玩家不能再『標示』或『翻開』某一格了，但是程式不見得要結束，一個視窗程式要不要結束不是程式該決定的，應該是玩遊戲的人該決定的，也就是應該要透過『遊戲/結束』的選單界面來決定。當某一次遊戲已經踩到地雷時，當然在畫面中要顯示出來，同時玩家如果繼續要運用滑鼠右鍵或是左鍵『標示』或『翻開』某一格時，程式應該不要理它，甚至給玩家一些訊息，跟他說這一次的遊戲已經結束了，不要再亂了。在我們前面談到的物件裡面，可以設計一個狀態變數來記錄對於某一次亂數產生的遊戲來說，到底是遊戲中還是遊戲已經結束，是贏了還是輸了，每一次『標示』以及『翻開』動作執行前要檢查這些狀態變數，決定要不要動作，如果執行動作的話，完畢以後要更新狀態。

```
while (遊戲尚未結束) {
    ❶ 讀取使用者指定的動作和座標點
    ❷ 顯示整個畫面
}
```

總結一下：『讀取使用者指定的動作和座標點』換成一個一個事件了，『遊戲是否結束』換成一些狀態變數的設定和檢查，結束不代表程式結束，而是程式不動作了，『顯示』的部份我們在下一步驟來談

16. 爲了讓你的程式可以在視窗界面中顯示資料和遊戲狀態，下面提供了 12 個函式，你應該只需要這些就可以完成整個程式的設計了：

void a_showFrawn(); 呼叫這個函式顯示哭臉 

void a_showSmile(); 呼叫這個函式顯示笑臉 

void a_showCell(int irow, int icol); 呼叫這個函式在 (irow,icol) 顯示

void a_showFlag(int irow, int icol); 在 (irow,icol) 顯示標示狀態 

void a_showBlastedMine(int irow, int icol);
在 (irow,icol) 顯示炸開的地雷 

void a_showMine(int irow, int icol); 在 (irow,icol) 顯示地雷 

void a_showQuestionMark(int irow, int icol); 在 (irow,icol) 顯示 

void a_showSurroundingMineCount(int count, int irow, int icol);
在 (irow,icol) 顯示   …

void a_setRemainingMineCount(int count); 顯示未標示地雷數 

void a_startClock(); 讓碼表開始讀秒 

void a_stopClock(); 停下碼表

void a_resetClock(); 重設碼表 

你可以隨便在哪一個函式裡先測試一下，例如呼叫
a_showFlag(1,2); a_showMine(5,7); a_startClock();

後記

- 這不是一個很偉大的程式，可是一個版本一個版本從最開始的
  - 純文字界面
  - 加上一些特殊符號的界面
  - 運用直接移動游標位置來避免在螢幕上顯示內容的閃動
  - 運用反白與上下左右鍵移動游標來標示位置，增加界面的方便性
  - 運用 non-blocking 的輸入方式使得使用者邊輸入，程式邊畫圖來豐富遊戲顯示的變化
  - 純粹事件驅動的視窗界面版本

  運用到不同於一般資料處理程式設計方法
- 如果你寒暑假有額外時間的話，也可以針對物件導向的程式設計以及 Win32 視窗的應用程式或是 Qt 的應用程式來學習