組合語言期末專題 - 數獨

資工二A 108502516 謝佞暟
資工二A 108502202 王國隴
光電二　108206524 郭兆霖
光電二　108206528 黃啓瑞

分工

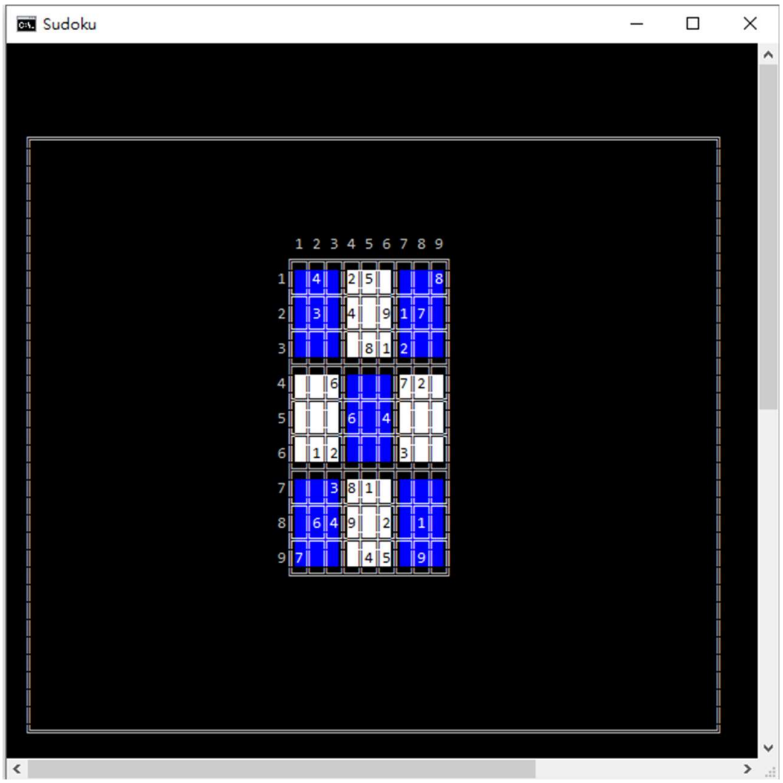
謝佞暟 : 數獨表格輸入，表格、版面刻劃
郭兆霖 : 後台邏輯判斷、儲存格式
黃啓瑞 : 程式整合、優化
王國隴 : 開始畫面設計

數獨操作說明

1. 進入開始畫面後，按下 Enter 鍵即可開始遊戲。

圖、開始畫面

2. 輸入 X 座標後按下 Enter 送出
3. 輸入 Y 座標後按下 Enter 送出
4. 輸入欲填入的數字後按下 Enter 送出，即可將數字填入想要的格子中。例如 : 輸入 X 座標 = 3，Y 座標 = 1，數字 = 9，結果呈現如下圖 :



圖、數獨遊戲畫面

5. 所有空格都填入正確數字後遊戲即結束。若填入的數字不是正確數字，則會用紅色字體提醒玩家輸入的數字為錯誤的。

函式庫

Irvine32

程式架構及流程

1. 印出遊戲開始畫面
   利用第 14 週上機實習的練習設計出開始畫面，先畫出外框，再畫出 SUDOKU 字樣，最後畫出 PLAY。例如 : 畫出 S 字樣，用變數儲存空白和米字號，再利用 **WriteConsoleOutputAttribute** 設定字體顏色，然後使用 **WriteConsoleOutputCharacter** 印出一行，接著將 **xyPosition** 的 **Y** 軸加 **1** 繼續印出下一行。其他字形依照此概念分別印出。

```
;   ******   '
; *        *
;*          *
;   *
;      *
;         *
;            *
;*           *
; *        *
;   ******
S1 BYTE 2 DUP(' '), 6 DUP('*'), 2 DUP(' ')
S2 BYTE ' ', '*', 6 DUP(' '), '*', ' '
S3 BYTE '*', 8 DUP(' '), '*'
S4 BYTE 2 DUP(' '), '*', 7 DUP(' ')
S5 BYTE 4 DUP(' '), '*', 5 DUP(' ')
S6 BYTE 6 DUP(' '), '*', 3 DUP(' ')
S7 BYTE 8 DUP(' '), '*', ' '
S8 BYTE '*', 8 DUP(' '), '*'
S9 BYTE ' ', '*', 6 DUP(' '), '*', ' '
S10 BYTE 2 DUP(' '), 6 DUP('*'), 2 DUP(' ')
```

圖、儲存字形

```
; 畫出 S
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S1, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S2, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S3, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S4, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S5, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S6, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S7, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S8, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S9, charWidth, xyPosition, ADDR cellsWritten
inc xyPosition.y
INVOKE WriteConsoleOutputAttribute, outputHandle, ADDR attributes_0, charWidth, xyPosition, ADDR cellsWritten
INVOKE WriteConsoleOutputCharacter, outputHandle, ADDR S10, charWidth, xyPosition, ADDR cellsWritten
```

圖、印出字體程式碼

2. 印出 9 * 9 表格及題目

表格分成4個部分

(1) [redacted] (2)[redacted]

(3)[redacted](4)[redacted]

第1行印出(1)，第2~17分別印出(2)、(3)，第18行印出(4)

第1、7、13、19列和行不更改顏色，每印完一列使印出位置向下1，持續到印完表格

```
    mov xyPosition.X, 32                    ;表格初始位置xy
    mov xyPosition.Y, 12

    INVOKE WriteConsoleOutputCharacter,     ;印出表格上半(此段不需要顏色設定)
      outputHandle,
      ADDR inputMapTop,
      inputMapLength,
      xyPosition,
      ADDR cellsWritten

    inc xyPosition.y                        ;換到下一行

    mov ecx, 2                              ;設定印出表格所需的迴圈數
inputMapStage :
    push ecx                                ;將counter存入stack避免其值被更動，影響迴圈

    INVOKE WriteConsoleOutputAttribute,     ;設定表格身體上部顏色
      outputHandle,
      ADDR attributes0,
      inputMapLength,
      xyPosition,
      ADDR cellsWritten

    INVOKE WriteConsoleOutputCharacter,     ;印出表格身體上部，此為第1、2列
      outputHandle,
      ADDR inputMapBody1,
      inputMapLength,
      xyPosition,
      ADDR cellsWritten

    inc xyPosition.y                        ;換到下一行

      INVOKE WriteConsoleOutputAttribute,   ;設定表格身體下部顏色
      outputHandle,
      ADDR attributes0,
      inputMapLength,
      xyPosition,
      ADDR cellsWritten

    INVOKE WriteConsoleOutputCharacter,     ;印出表格身體下部
      outputHandle,
      ADDR inputMapBody2,
      inputMapLength,
      xyPosition,
      ADDR cellsWritten

    inc xyPosition.y                        ;換到下一行
    pop ecx                                 ;將原本存入的counter提出，保持其計算正確
    dec ecx                                 ;迴圈次數-1
    cmp ecx, 0                              ;counter與0比對，如果不同則跳回印出表格身體階段
    jne inputMapStage                       ;由於loop 無法跳躍過長的距離，改以jmp的方式進行
```

圖、第1~5列列印

印完表格後，將印出位置調整到表格<1, 1>的上側，由於陣列一開始就設好，可以直接印出上標頭
再來將印出位置調整到表格<1, 1>左側，由於沒辦法一次印完，設置迴圈9次，分別印出9個數字，印出陣列引數每次要加二，因為印出標頭的陣列每個數字有用空白隔開

```
    mov xyPosition.x, 33
    mov xyPosition.y, 11
    INVOKE WriteConsoleOutputCharacter,      ;印出上標頭，由於初始點為<33,13>，將Y-2為上標頭起始
      outputHandle,
      ADDR header,
      lengthof header,
      xyPosition,
      ADDR cellsWritten

    mov edi, offset header

    mov xyPosition.x, 31
    mov xyPosition.y, 13
    mov ecx, 9                               ;1~9共9個數字
printHeadLeft:
    push ecx
    INVOKE WriteConsoleOutputCharacter,      ;印出左標頭，由於初始點為<33,13>，將X-2為左標頭起始
      outputHandle,
      edi,
      1,
      xyPosition,
      ADDR cellsWritten
    add edi, 2                               ;字串中有空白隔開，所以加二才是下一個數字
    add xyPosition.y, 2                      ;Y往下2才是往下一個表格位置
    pop ecx
    loop printHeadLeft
```

圖、上標頭和左標頭列印

接下來判斷題目的81個位置是否為題目並印出，前置先設定好表格值、顏色、答案、題目判斷的地址，並將印出位置調整到表格<1, 1>

```
    mov ecx, 81                              ;設定要印出題目的判斷迴圈數
    mov edi, offset mapValue                 ;預設好表格值地址存入mapValueIndex
    mov mapValueIndex, edi
    mov edi, offset mapColor                 ;預設好表格顏色地址存入mapColorIndex
    mov mapColorIndex, edi
    mov edi, offset taskAnswer               ;題目答案的array地址
    mov esi, offset mapValueBool             ;是否印出數值的array地址
    mov xyCur.x, 33                          ;設定表格第一格xy位置
    mov xyCur.y, 13
```

圖、印出題目的前置

當判斷到這一格為題目時，將印出位置移至此格，判斷此格位置的顏色應該為藍或白並設定，並將其印出，由於之後需要連同題目一起判定這張數獨是否完成，所以要把題目的值存入到表格值的陣列中，由於題目中存入的數字為字元，需將其減去'0'變成數值，才能存入表格值中

```
printTask:
    mov edx, 0                                  ;清空edx
    mov dl, [esi]                               ;將此格題目印出
    .IF dl == '1'                               ;如果此格為題目，也就是判斷為'1'
        ;設定位置到dh(x), dl(y)，設定印出位置
        mov eax, 0
        mov ax, xyCur.x
        mov dl, al
        mov ax, xyCur.y
        mov dh, al
        call gotoxy
        ;此段為判斷該格的顏色，先將edi存入stack保持其值，然後設定印出顏色
        push edi
        mov edi, mapColorIndex
        mov eax, 0
        mov al, [edi]
        .IF al == '0'
            mov eax, white+(lightblue*16)
        .ENDIF
        .IF al == '1'
            mov eax, black+(white*16)
        .ENDIF
        call SetTextColor
        pop edi
        ;將題目存入al後印出
        mov al, [edi]
        call writechar
        ;將題目的值存入到表格值，之後判斷是否題目完成用
        push edi
        mov edi, mapValueIndex
        sub al, '0'                             ;減掉'0'會使其變成數值
        mov ebx, 0
        mov bl, al
        mov BYTE ptr [edi], bl
        pop edi
    .ENDIF
```

圖、印出題目的各種處理

表格中的位置，每往右一格、X加2，往下一格、Y加2，在每次的迴圈中，向右一格，如果超過49，表示超過每一列最後一格，則換到下一行第一格，接下來把所有需要用到的位址只到下一位，分別為答案、表格值、顏色、題目判斷

```
mov ax, xyCur.x                          ;存入此格的x位置後將其指到右邊一格
add ax, 2
.IF ax > 49
    mov ax, 33
    mov bx,xyCur.y
    add bx, 2
    mov xyCur.y, bx
.ENDIF
mov xyCur.x, ax
;答案、表格內的值、顏色判斷和題目判斷array指向下一位
inc edi
push edi
mov edi, mapValueIndex
inc edi
mov mapValueIndex, edi
pop edi
push edi
mov edi, mapColorIndex
inc edi
mov mapColorIndex, edi
pop edi
inc esi
dec ecx
cmp ecx, 0
jne printTask                            ;loop直到所有表格內容都判斷並印出
```

圖、印出位置移動和各種引數增加

3. 使用者輸入 X 座標、Y 座標、數值

接下來會一直迴圈直到數獨完成或輸入10跳出程式，首先輸入X的值，如果小於9則存入，等於10則結束遊戲，輸入完後將游標重置到左上，清空剛才的輸入

```
;此段為判斷跳出程式和user指定表格x位置，輸入完後重置游標和清除輸入
mov eax, 0
call readint
.IF eax <= 9
    mov xyPos.x, ax                      ;在xyPos中存入user輸入表格的X位置
.ENDIF
.IF eax == 10
    jmp END_stage
.ENDIF
INVOKE WriteConsoleOutputCharacter,
   outputHandle,
   ADDR clearBuffer,
   lengthof clearBuffer,
   xyInit,
   ADDR cellsWritten
INVOKE SetConsoleCursorPosition, consoleHandle, xyInit
```

圖、輸入表格第幾行(X位置)

Y的輸入和X的輸入差不多，不過少了跳出判斷

```
;此段為user指定表格y位置，輸入完後重置游標和清除輸入
call readint
.IF eax <= 9
    mov xyPos.y, ax                    ;在xyPos中存入user輸入表格的Y位置
.ENDIF
INVOKE WriteConsoleOutputCharacter,
   outputHandle,
   ADDR clearBuffer,
   lengthof clearBuffer,
   xyInit,
   ADDR cellsWritten
INVOKE SetConsoleCursorPosition, consoleHandle, xyInit
```

圖、輸入表格第幾列(Y位置)

接下來輸入USER欲存入的數值，如果輸入的值小於9，先根據輸入的X、Y更改index，做為等下存入用，index的規則為每九個一列，因此index加1代表向右一格，加9代表向下一格，由於USER輸入的是位置，在進行引數處理時，需要先將X和Y的值減一，然後Y的值乘以9，相加後即是表格值的index

找到index後，分別將表格值、題目判斷、顏色指向這個位置，如果判斷這一格不是題目，則將其值存入表格值中，並判斷這個數獨是否完成，如果完成了就結束遊戲

```
;此段將user存入表格的數值紀錄下來
call readint
.IF eax <= 9
    mov mapTemp, al                         ;mapTemp存入user輸入的值
    mov edi, offset mapValue
    mov eax, 0
    mov ebx, 0
    ;使用者輸入的y位置，將其減一並乘以9(由於陣列為9個一列，引數每加9就向下一列)
    mov ax, xyPos.y
    dec ax
    mov bx, 9
    mul bx
    ;使用者輸入的x位置，將其減一
    mov bx, xyPos.x
    dec bx
    add ax, bx
    ;處理過後的x和y相加即是引數，用於表格值和顏色判斷
    mov mapIndex, ax
    mov ebx, 0
    mov bx, mapIndex
    add edi, ebx
    mov esi, offset mapValueBool
    add esi, ebx
    push esi
    mov esi, offset mapColor
    add esi, ebx
    mov mapColorIndex, esi
    pop esi
    mov al, [esi]
    ;如果al為'0'，代表這題為user可輸入的地方
    .IF al == '0'
        mov ebx, 0
        mov bl, mapTemp
        mov BYTE ptr [edi], bl
    .ENDIF
    ;判斷表格內所有的值是否為正確
    INVOKE judgeRow
    .IF al == 1
        jmp END_stage
    .ENDIF
    INVOKE judgeColumn
    .IF al == 1
      jmp END_stage
    .ENDIF
    INVOKE judgeCube
    .IF al == 1
      jmp END_stage
    .ENDIF

.ENDIF
```

圖、USER數值輸入(上)

這段為輸入位置到USER輸入的表格位置的前置處理，方法是先將USER輸入的X和Y減一，並將其乘以2(每往右或下要加2才是下一格)，然後加上表格<1, 1>的位置

```
;指定表格的位置後將user指定的值填入表格
mov eax, 0
mov ax, xyPos.x
dec eax
mov ecx, 2
mul ecx
mov ebx, 0
mov bx, xyInit2.x
add ebx, eax
mov xyCur.x, bx

mov eax, 0
mov ax, xyPos.y
dec eax
mov ecx, 2
mul ecx
mov ebx, 0
mov bx, xyInit2.y
add ebx, eax
mov xyCur.y, bx
```

圖、USER數值輸入(中)

接下來把印出位置改到剛剛處理好的位置，根據這個位置的顏色判斷更改背景顏色，並根據USER輸入的數字，判定這一格輸入的是不是答案，如果是文字改成綠色，不是則文字改成紅色，由於現在的輸入值是數值，將其 or 00110000b 可以變成字元，就能夠印出，最後將輸入、游標、顏色重置

```
;如果這一格是題目，則略過
mov eax, 0
mov al, [esi]
.IF al == '0'
    mov eax, 0
    mov ax, xyCur.x
    mov dl, al
    mov ax, xyCur.y
    mov dh, al
    call gotoxy

    push esi
    mov esi, mapColorIndex
    mov eax, 0
    mov al, [esi]
    .IF al == '0'
        mov eax, lightblue*16
    .ENDIF
    .IF al == '1'
        mov eax, white*16
    .ENDIF
    push esi
    mov esi, offset taskAnswer
    mov ebx, 0
    mov bx, mapIndex
    add esi, ebx
    mov ebx, 0
    mov bl, mapTemp
    add bl, '0'
    push ecx
    mov ecx, 0
    mov cl, [esi]
    .IF bl == cl
        add eax, lightgreen
    .ENDIF
    .IF bl != cl
        add eax, red
    .ENDIF
    pop ecx
    pop esi

    call SetTextColor
    mov al, mapTemp
    or al, 00110000b
    call writechar
.ENDIF
INVOKE SetConsoleCursorPosition, consoleHandle, xyInit
mov eax, white+(black*16)
call SetTextColor

jmp L2
```

圖、USER數值輸入(下)

4. 判斷輸入的所有數值是否正確

先以列的判斷為例：

首先數獨總共有9列，所以需要跑9圈迴圈
先將儲存數字使用次數的記憶體清空

```
judgeRow PROC USES ebx ecx esi edi
    mov eax, 1
    mov edi, OFFSET mapValue
    mov ecx, 9
    rows:
    mov esi, OFFSET contentJudge

    push ecx
    mov ecx, 9
        clear:
        mov bl, 0
        mov [esi], bl
        add esi, 1
        loop clear
```

將數字的出現次數計入

```
1059        mov esi, OFFSET contentJudge
1060        mov ecx, 9
1061            judge:
1062            push ecx
1063            mov ecx, 9
1064                oneOf:
1065                push esi
1066                cmp [edi], cl
1067                jne notDo
1068                add esi, ecx
1069                dec esi
1070                mov bl, [esi]
1071                inc bl
1072                mov [esi], bl
1073                notDo:
1074                pop esi
1075                loop oneOf
1076            add edi, 1
1077            pop ecx
1078            loop judge
```

判斷這一列是否所有數字數量都為1
若不是，那必定會有數字數量為0，將其相乘結果也為0

```
1080        mov ecx, 9
1081            rowIs:
1082            mov bl, [esi]
1083            mul bl
1084            inc esi
1085            loop rowIs
1086
1087        pop ecx
1088        loop rows
1089        ret
1090        ;最終結果存在eax(0錯1對)
1091    judgeRow ENDP
```

其餘兩者原理相同，在此不重複解釋

```
judgeColumn PROC USES ecx esi edi
judgeCube PROC USES ecx esi edi
```