組合語言與系統程式 **Final Project**

Group: 17

109504501曾千芸 **110502514**賈子悅 **110201520**張業僖 **109201201**吳峻凱

- **程式架構(function)**

  print PROTO

  Transfer PROTO, X1:sbyte, Y1:sbyte

  Boundary PROTO, x:sbyte, y:sbyte

  movecursor PROTO

  Choose PROTO

  movechess PROTO

  Iswin PROTO

  IsChess PROTO, boolx:sbyte, booly:sbyte

  COOR STRUCT

  X sbyte 0

  Y sbyte 0

  COOR ENDS

  explaination:

  print :繪出棋盤

  Transfer :棋盤座標與小黑窗座標轉換

  Boundary:判斷輸入的座標是否在棋盤內

  movecursor:讓玩家選擇要移動的棋子

  Choose:鎖定要移動的棋子

  movechess:移動棋子

  Iswin:判斷是否有棋子全部移動至對角

  IsChess:判斷使用者是否選對到棋子，而不是空格或對手的棋子

  COOR STRUCT :

  X sbyte 0

  Y sbyte 0

COOR ENDS

一個struct, 用來儲存座標

- **程式架構(data)**

;棋子資料

B COOR <-4, 8>, <-4, 7>,.....

M COOR <-4, 0>, <-4,-1>,.....

Br COOR < 4, 0>, < 4,-1>,.....

cursor COOR < 0, 0>

dot byte "o"

control byte 1

explaination:

B ,M ,Br:用上述位置的結構儲存每個顏色的15隻棋位置

cursor COOR < 0, 0>:一個用來存放目前游標位置的結構

dot :設定棋盤的樣式

control :設定目前遊戲玩家是誰, 1代表藍, 2代表紫, 3代表黃

;遊戲狀態訊息

```
chos   byte "choose",0
unchos byte "unlock",0
En     byte "end",0
bluewin, magentawin, brownwin byte
Jumporstop byte
havechess   byte
cantjump    byte
chooseyourschess byte
introduction byte
introduction2 byte
introduction3 byte
way_to_play byte
```

explaination:

chos :提示使用者選取了棋子

unchos:提示使用者取消選取了棋子

En:提示使用者結束這一步
bluewin, magentawin, brownwin:提示某一使用者獲勝
Jumporstop:提示使用者是否繼續往下跳
Haveches:提示使用者要跳的位置有其他棋子
Cantjump:提示使用者該位置不能跳
Chooseyourschess:提示使用者要選自己的棋子
introduction, introduction2, introduction3:遊戲介紹
way_to_play:玩法介紹

- 使用函式庫

	main	print	movecursor	movechess	Choose
clrscr		✓			
Gotoxy	✓	✓		✓	✓
readchar	✓		✓		
SetTextColor		✓			
Writechar		✓			
WriteString		✓		✓	✓

- Screenshot of program

```
;跳棋程式
INCLUDE Irvine32.inc

;繪出棋盤
print PROTO

;棋盤座標與小黑窗座標轉換
Transfer PROTO,
    X1:sbyte,
    Y1:sbyte

;判斷輸入的座標是否在棋盤內
Boundary PROTO,
    x:sbyte,y:sbyte

;判斷有無玩家勝利
Iswin PROTO

;判斷輸入的位置是否有棋子
IsChess PROTO,
    boolx:sbyte,booly:sbyte

;選擇棋子
Choose PROTO

;移動游標
movecursor PROTO

;移動棋子
movechess PROTO

;棋子座標結構(帶正負的座標)
COOR STRUCT
    X sbyte 0
    Y sbyte 0
COOR ENDS
```

```
    .data
    ;15集蓝棋位置
    B COOR <-4, 8>, <-4, 7>, <-3, 7>, <-4, 6>, <-3, 6>, <-2, 6>, <-4, 5>, <-3, 5>, <-2, 5>, <-1, 5>, <-4, 4>, <-3, 4>, <-2, 4>, <-1, 4>, < 0, 4>
    ;15集红紫棋位置
    M COOR <-4, 0>, <-4,-1>, <-3,-1>, <-4,-2>, <-3,-2>, <-2,-2>, <-4,-3>, <-3,-3>, <-2,-3>, <-1,-3>, <-4,-4>, <-3,-4>, <-2,-4>, <-1,-4>, < 0,-4>
    ;15集咖啡棋位置
    Br COOR < 4, 0>, < 4,-1>, < 5,-1>, < 4,-2>, < 5,-2>, < 6,-2>, < 4,-3>, < 5,-3>, < 6,-3>, < 7,-3>, < 4,-4>, < 5,-4>, < 6,-4>, < 7,-4>, < 8,-4>

    ;快赢的15集蓝棋位置
    ;B COOR <4, -8>, <4, -7>, <3, -7>, <4, -6>, <3, -6>, <2, -6>, <4, -5>, <2, -3>, <2, -5>, <1, -5>, <4, -4>, <3, -4>, <2, -4>, <1, -4>, < 0, -4>
    ;快赢的15集红紫棋位置
    ;M COOR <4, 0>, <4,1>, <4,2>, <4,3>, <4,4>, <3,0>, <3,2>, <3,3>, <3,4>, <2,2>, <2,3>, <2,4>, <1,3>, <1,4>, < 0,4>
    ;快赢的15集咖啡棋位置
    ;Br COOR <-4, 0>, <-4,1>, <-4,2>, <-4,3>, <-4,4>, <-3,1>, <-5,2>, <-5,3>, <-5,4>, <-6,2>, <-6,3>, <-6,4>, <-7,3>, <-7,4>, <-8,4>

    ;棋盘样式
    dot byte "o"

    ;游标位置
    cursor COOR < 0, 0>

    ;控制權
    control byte 1

    ;遊戲狀態
    chos   byte "choose",0
    unchos byte "unlock",0
    En     byte "end",0
    bluewin     byte "The winner is blue!!!",0
    magentawin  byte "The winner is magenta!!!",0
    brownwin    byte "The winner is yellow!!!",0
    jumporstop  byte "Choose another position to arrive, or press Enter again to stop the chess here. ",0
    havechess   byte "This position has a chess. ",0
    cantjump    byte "You can't jump to the position",0
    chooseyourschess byte "You Can't choose other's chess or choose Nothing. Please choose your own chess! ",0

    introduction byte "Sternhalma(Chinese checkers)",0
    introduction2 byte "is a strategy board game of German origin that can be played by two, three, four, or six people,",0
    introduction3 byte "playing individually or with partners. The game is a modern and simplified variation of the game Halma.",0
    way_to_play byte "use 'up' 'down' 'left',and 'right' to control, 'enter' to select. ",0
```

```
.code
main PROC

        INVOKE print                            ;印出初始棋盤

    gameconti:

        INVOKE Iswin                            ;判斷有無玩家勝利
        cmp  ax,1
        jz somebodywin                          ;有玩家勝利
        INVOKE Choose                           ;選擇棋子
        mov  edx, OFFSET chos
        call WriteString                        ;輸出"choose"狀態
        INVOKE movechess                        ;移動棋子
        cmp  ax, 0
        jz   unchoose                           ;重新選擇棋子
        INVOKE print                            ;繪出棋盤
        mov  edx, OFFSET En
        call WriteString                        ;輸出"end"狀態

        add control, 1                          ;control = (control + 1) % 3 + 1
        cmp control, 3
        jng gameconti
        add control, -3
        mov al,control
        jmp gameconti                           ;遊戲繼續

    unchoose:                                   ;重新選擇棋子
        INVOKE print                            ;繪出棋盤
        mov  edx, OFFSET unchos
        call WriteString                        ;輸出"unlock"狀態
        jmp  gameconti

    somebodywin:                                ;有玩家勝利
        cmp  bx,1
        jz   wblue
        cmp  bx,2
        jz   wmagenta
        cmp  bx,3
        jz   wbrown

    wblue:                                      ;藍棋勝利
        mov  dh, 0
        mov  dl, 0
        call Gotoxy
        mov  edx, OFFSET bluewin                    ;印出"The winner is blue!!!"狀態
        call WriteString
        jmp  endgame

    wmagenta:                                       ;紅紫棋勝利
        mov  dh, 0
        mov  dl, 0
        call Gotoxy
        mov  edx, OFFSET magentawin             ;印出"The winner is magenta!!!"狀態
        call WriteString
        jmp  endgame

    wbrown:                                 ;咖啡棋勝利
        mov  dh, 0
        mov  dl, 0
        call Gotoxy
        mov  edx, OFFSET brownwin               ;印出"The winner is brown!!!"狀態
        call WriteString
        jmp  endgame

    endgame:                                    ;遊戲結束
        call readchar
        exit

main ENDP
```

```
print PROC

        call clrscr                             ;清空螢幕

        push eax                                ;將會用到的暫存器內容放入堆疊
        push ebx
        push ecx
        push esi

    ;---繪製大棋盤白點---
    ;雙層迴圈 -8 <= i,j <= 8
    ;透過 Boundary 以及透過 Transfer 轉換，繪出棋盤上的點出棋盤上的點
        mov  ecx, 17
    printintro:

        mov  al , 15
        call SetTextColor
        mov  dh , 1
        mov  dl , 0
        call Gotoxy
        mov  edx, OFFSET introduction
        call WriteString
        mov  dh , 2
        mov  dl , 0
        call Gotoxy
        mov  edx, OFFSET introduction2
        call WriteString
        mov  dh , 3
        mov  dl , 0
        call Gotoxy
        mov  edx, OFFSET introduction3
        call WriteString
        mov  al , 3
        call SetTextColor
        mov  dh , 27
        mov  dl , 0
        call Gotoxy
        mov edx, OFFSET may_to_play
        call WriteString
    outter:
        mov   bh , cl
        sub   bh , 9

        push  ecx

        mov   ecx, 17
    inner:
        mov   bl , cl
        sub   bl , 9
        INVOKE Boundary, bh, bl
        cmp  ax , 0
        jz   outofbound                         ;點出界 --> 不印出

        INVOKE Transfer, bh, bl
        call Gotoxy                             ;游標位置已經由 Transfer 算好，並放入遠當的暫存器 (dl,dh)
        mov  eax, white + ( black*16 )          ;設定前景為白色，背景為黑色
        call SetTextColor

        mov  al , dot
        call Writechar                          ;印出點點

    outofbound:

        loop inner

        pop ecx

        loop outter
```

```
        ;---大棋盤紅點---
        ;按照矩陣的內容, 透過 Transfer 轉換後繪出
            mov  ecx, 15
            mov  esi, OFFSET B

        printblue:
            mov  bh , (COOR PTR [esi]).X
            mov  bl , (COOR PTR [esi]).Y
            INVOKE Transfer, bh, bl
            call Gotoxy
            mov  eax, 1 + ( black*16 )          ;設定前景為藍色, 背景為黑色
            call SetTextColor

            mov  al , 3h
            call Writechar                      ;印出愛心

            add  esi, TYPE COOR
            loop printblue

        ;---大棋盤紅紫點---
        ;按照矩陣的內容, 透過 Transfer 轉換後繪出
            mov  ecx, 15
            mov  esi, OFFSET M


        printmagenta:
            mov  bh , (COOR PTR [esi]).X
            mov  bl , (COOR PTR [esi]).Y
            INVOKE Transfer, bh, bl
            call Gotoxy
            mov  eax, 5 + ( black*16 )          ;設定前景為紅紫色, 背景為黑色
            call SetTextColor

            mov  al , 6h
            call Writechar                      ;印出黑桃

            add  esi, TYPE COOR
            loop printmagenta

        ;---大棋盤咖啡點---
        ;按照矩陣的內容, 透過 Transfer 轉換後繪出
            mov  ecx, 15
            mov  esi, OFFSET Br

        printbrown:
            mov  bh , (COOR PTR [esi]).X
            mov  bl , (COOR PTR [esi]).Y
            INVOKE Transfer, bh, bl
            call Gotoxy
            mov  eax, 6 + ( black*16 )      ;設定前景為咖啡色, 背景為黑色
            call SetTextColor

            mov  al , 4h
            call Writechar                      ;印出菱形
            add  esi, TYPE COOR
            loop printbrown
```

```
        ;---大棋盤游標---
            mov  bh , cursor.X
            mov  bl , cursor.Y
            INVOKE Transfer, bh, bl

            ;繪製"[" , 淡青綠色
            sub  dl ,1
            call Gotoxy
            mov  eax, 11 + ( black*16 )
            call SetTextColor
            mov  al , "["
            call Writechar

            ;繪製"[" , 淡青綠色
            add  dl ,2
            call Gotoxy
            mov  eax, 11 + ( black*16 )
            call SetTextColor
            mov  al , "]"
            call Writechar

        ;從堆疊取回暫存器內容
            pop  esi
            pop  ecx
            pop  ebx
            pop  eax

            ret

    print ENDP
```

```
;-----------------------------------------------Transfer-----------------------------------------------
Transfer PROC,
    X1:sbyte, Y1:sbyte

    ;將會用到的暫存器內容放入堆疊
    push eax
    push ebx

    mov al, X1;
    mov bl, Y1;

    ;2*x1 + y1 + 13 =x2

    add al, al
    add al, bl
    add al, 30
    mov dl, al

    ;-y1 + 9 = y2
    neg bl
    add bl, 15
    mov dh, bl

    ;從堆疊取回暫存器內容
    pop ebx
    pop eax

    ret

Transfer ENDP
;-----------------------------------------------Boundary-----------------------------------------------
Boundary PROC, ux:SBYTE, uy:SBYTE
        push ebx
    test_uptri:

        ;判斷是否在上三角裡，只要有一項不符合，
        ;就直接跳去判斷是不是下三角里
        cmp uy, -4
        jl test_downtri  ;y<-4
        cmp ux, -4
        jl test_downtri  ;x<-4
        mov bl, ux
        add bl, uy
        cmp bl, 4
        jg test_downtri  ;x+y>4
        jmp Istrue

        ;三項都符合，所以在上三角裡
        ;就直接跳去Istrue，回傳true(ax=1)，結束程式

    test_downtri:
        ;判斷是否在下三角裡，只要有一項不符合，
        ;就直接跳去Isfalse，回傳false(ax=0)，結束程式
        cmp uy, 4
        jg Isfalse  ;y > 4
        cmp ux, 4
        jg Isfalse  ;x > 4
        mov bl, ux
        add bl, uy
        cmp bl, -4
        jl Isfalse  ;x+y>=-4

        ;三項都符合，所以在下三角裡就
        ;到Istrue，回傳true(ax=1)，結束程式

    Istrue:
        mov ax, 1
        jmp existBoundary
    Isfalse:
        mov ax, 0
    existBoundary:
        pop ebx
        ret

Boundary ENDP
```

```
Iswin PROC
        push edi
        push ecx
        mov ecx, 15
        mov edi, 0
    checkR:
        cmp (COOR PTR B[edi]).Y, -4         ;B的每個棋子的Y都必須小於等於-4
        jg checkM                           ;只要有一顆沒有，R就不可能贏，就直接跳去check M
        cmp (COOR PTR B[edi]).X, 0          ;B的每個棋子的X都必須大於等於0
        jl checkM                           ;只要有一顆沒有，B就不可能贏，就直接跳去check M
        cmp (COOR PTR B[edi]).X, 4          ;B的每個棋子的X都必須小於等於4
        jg checkM                           ;只要有一顆沒有，B就不可能贏，就直接跳去check M
        add edi, TYPE COOR
        loop checkR
        mov bx, 1
        jmp Win                             ;B每顆棋子的Y都小於等於-4，則B贏了

    checkM:
        mov ecx, 15
        mov edi, 0
    Mloop:
        mov bl, (COOR PTR M[edi]).X         ;M的每個棋子都必須x+y>=4
        add bl, (COOR PTR M[edi]).Y
        cmp bl, 4
        jl checkBr                          ;只要有一顆沒有，M就不可能贏，就直接跳去check Br
        cmp (COOR PTR M[edi]).Y, 0          ;M的每個棋子的Y都必須大於等於0
        jl checkBr                          ;只要有一顆沒有，M就不可能贏，就直接跳去check Br
        cmp (COOR PTR M[edi]).Y, 4          ;M的每個棋子的Y都必須小於等於4
        jg checkBr                              ;只要有一顆沒有，M不可能贏，就直接跳去check Br
        add edi, TYPE COOR
        loop Mloop
        mov bx, 2
        jmp Win                             ;M的每個棋子都x+y>=4，則M贏了

    checkBr:
        mov ecx, 15
        mov edi, 0
    Brloop:
        cmp (COOR PTR Br[edi]).X, -4            ;Br的每個棋子的X都必須小於等於-4
        jg Conti                            ;只要有一顆沒有，Br就不可能贏，即B,M,Br都沒有人贏
        cmp (COOR PTR Br[edi]).Y, 0         ;Br的每個棋子的Y都必須大於等於0
        jl Conti                            ;只要有一顆沒有，Br就不可能贏，即B,M,Br都沒有人贏 Y
        cmp (COOR PTR Br[edi]).Y, 4         ;Br的每個棋子的Y都必須小於等於4
        jg Conti                            ;只要有一顆沒有，Br就不可能贏，即B,M,Br都沒有人贏
        add edi, TYPE COOR
        loop Brloop
        mov bx, 3
        jmp Win                             ;Br每顆棋子的X都小於等於-4，則Br贏了

    Win:                                    ;有玩家勝利 ax = 1
        mov ax, 1
        jmp existIswin

    Conti:                                  ;無玩家勝利 ax = 0
        mov ax, 0;

    existIswin:
        pop ecx
        pop edi
        ret

Iswin ENDP
```

```
movecursor PROC

        push eax
        push ebx

    ;等待輸入上、下、左、右、enter
    WaitInput:
        call readchar
        cmp  eax, 4800h                 ;上
        jz   UP
        cmp  eax, 5000h                 ;下
        jz   DOWN
        cmp  eax, 4B00h                 ;左
        jz   LEFT
        cmp  eax, 4D00h                 ;右
        jz   RIGHT
        cmp  eax, 1C0Dh                 ;enter
        jz   OUTFUN
        jmp  WaitInput

    ;上 -> Y座標+1 並判斷位置是否超出邊界
    UP:
        mov  bh, cursor.X
        mov  bl, cursor.Y
        add  bl, 1
        INVOKE Boundary, bh, bl
        cmp  ax , 0
        jz   WaitInput
        mov  cursor.X, bh
        mov  cursor.Y, bl
        INVOKE print
        jmp  WaitInput

    ;下 -> Y座標-1 並判斷位置是否超出邊界
    DOWN:
        mov  bh, cursor.X
        mov  bl, cursor.Y
        add  bl, -1
        INVOKE Boundary, bh, bl
        cmp  ax , 0
        jz   WaitInput
        mov  cursor.X, bh
        mov  cursor.Y, bl
        INVOKE print
        jmp  WaitInput

    ;左 -> X座標-1 並判斷位置是否超出邊界
    LEFT:
        mov  bh, cursor.X
        mov  bl, cursor.Y
        add  bh, -1
        INVOKE Boundary, bh, bl
        cmp  ax , 0
        jz   WaitInput
        mov  cursor.X, bh
        mov  cursor.Y, bl
        INVOKE print
        jmp  WaitInput

    ;右 -> X座標+1 並判斷位置是否超出邊界
    RIGHT:
        mov  bh, cursor.X
        mov  bl, cursor.Y
        add  bh, 1
        INVOKE Boundary, bh, bl
        cmp  ax , 0
        jz   WaitInput
        mov  cursor.X, bh
        mov  cursor.Y, bl
        INVOKE print
        jmp  WaitInput

    ;enter 跳出函式
    OUTFUN:
        pop  ebx
        pop  eax
        ret

movecursor ENDP
```

```
;-----------------------------------------------movechess-----------------------------------------------
movechess PROC

        ;宣告區域變數  isjmp:判斷是否為"跳"棋  chessx:棋子X座標  chessy:棋子Y座標
        LOCAL isjmp : byte, chessx : sbyte,chessy : sbyte
        push ebx
        push edx

        ;isjmp預設為0
        mov  isjmp, 0

    moveagain:

        ;移動游標
        INVOKE movecursor

        ;將棋子原始位置寫入 (bh, bl)
        mov  bh, (COOR PTR [esi]).X
        mov  chessx, bh
        mov  bl, (COOR PTR [esi]).Y
        mov  chessy, bl

        ;檢查游標位置是否有棋子
        INVOKE IsChess, cursor.X, cursor.Y
        cmp  al, 0
        jz   startmove                      ;該位置無棋子

        ;如果(bh, bl) == (cursor.X, cursor.Y) => 處理跳棋結束 或者 取消選取
        cmp  bh, cursor.X
        jnz  haschess
        cmp  bl, cursor.Y
        jnz  haschess
        jmp  startmove

    haschess:
        mov  dh, 0
        mov  dl, 0
        call Gotoxy
        mov  edx, OFFSET havechess          ;印出"This position has a chess."狀態
        call WriteString
        jnz  moveagain

    startmove:
        ;(bh, bl) 紀錄後來位置與原始位置的差
        mov  bh, cursor.X
        mov  bl, cursor.Y
        sub  bh, chessx
        sub  bl, chessy

        ;跳過的棋子不用判斷移動
        cmp  isjmp, 0
        jnz  jump                           ;處理"跳"棋

        cmp  bh, 1
        jz   xplusone                       ;X座標差 = +1(走、跳)
        cmp  bh, 0
        jz   xremain1                       ;X座標差 = 0 (走、跳、取消選取)
        cmp  bh, -1
        jz   xminusone                      ;X座標差 = -1(走、跳)
        cmp  bh, 2
        jz   xplustwo                       ;X座標差 = +2(跳)
        cmp  bh, -2
        jz   xminustwo                      ;X座標差 = -2(跳)
        jmp  invalidmove                    ;X座標差超出範圍(非法移動)

    ;X座標差 = +1(走、跳)
    xplusone:
        cmp  bl, 0
        jz   moveright;                     ;Y座標差 = 0 (向右走)
        cmp  bl, -1
        jz   moverightdown                  ;Y座標差 = -1(向右下走)
        jmp  invalidmove                    ;Y座標差超出範圍(非法移動)
```

```
598         ;向右走
599         moveright:
600             mov  bh, chessx
601             add  bh, 1
602             mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
603             INVOKE print
604             mov  ax, 1
605             pop  ebx
606             ret
607
608         ;向右下走
609         moverightdown:
610             mov  bh, chessx
611             add  bh, 1
612             mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
613             mov  bl, chessy
614             add  bl, -1
615             mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
616             INVOKE print
617             mov  ax, 1
618             pop  ebx
619             ret
628
621         ;X座標差 = 0 (走、跳、取消選取)
622         xremain1:
623             cmp  bl, 1
624             jz   moverightup                    ;Y座標差 = 1 (向右上走)
625             cmp  bl, -1
626             jz   moveleftdown                   ;Y座標差 = -1 (向左下走)
627             cmp  bl, 0
628             jz   unlock                         ;Y座標差 = 0 (取消選取)
629             cmp  bl, 2
630             jz   jumprightup                    ;Y座標差 = 2 (向右上跳)
631             cmp  bl, -2
632             jz   jumpleftdown                   ;Y座標差 = -2 (向左下跳)
633             jmp  invalidmove                    ;Y座標差超出範圍(非法移動)
634
635         ;取消選取
636         unlock:
637             mov  ax, 0                          ;設定 ax = 0 供主程式判斷控制權轉移
638             pop  ebx
639             ret
640
641         ;向右上走
642         moverightup:
643             mov  bh, chessx
644             mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
645             mov  bl, chessy
646             add  bl, 1
647             mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
648             INVOKE print
649             mov  ax, 1
650             pop  ebx
651             ret
652
653         ;向左下走
654         moveleftdown:
655             mov  bh, chessx
656             mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
657             mov  bl, chessy
658             sub  bl, 1
659             mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
660             INVOKE print
661             mov  ax, 1
662             pop  ebx
663             ret
664
665         ;X座標差 = -1 (走、跳)
666         xminusone:
667             cmp  bl, 0
668             jz   moveleft                       ;Y座標差 = 0 (向左走)
669             cmp  bl, 1
670             jz   moveleftup                     ;Y座標差 = 1 (向左上走)
671             jmp  invalidmove                    ;Y座標差超出範圍(非法移動)
```

```
        ;向左走
        moveleft:
            mov  bh, chessx
            add  bh, -1
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            INVOKE print
            mov  ax, 1
            pop  ebx
            ret

        ;向左上走
        moveleftup:
            mov  bh, chessx
            add  bh, -1
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            mov  bl, chessy
            add  bl, 1
            mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
            INVOKE print
            mov  ax, 1
            pop  ebx
            ret

        ;這裡處理"跳"棋
        jump:
            cmp  bh, 2
            jz   xplustwo                       ;X座標差 = +2(跳)
            cmp  bh, 0
            jz   xremain2                       ;X座標差 = 0 (跳)
            cmp  bh, -2
            jz   xminustwo                      ;X座標差 = -2(跳)
            jmp  invalidmove                    ;X座標差超出範圍(非法移動)

        xplustwo:
            cmp  bl, 0
            jz   jumpright                      ;Y座標差 = 0 (向右跳)
            cmp  bl, -2
            jz   jumprightdown                  ;Y座標差 = -2 (向右下跳)
            jmp  invalidmove                    ;Y座標差超出範圍(非法移動)

        jumpright:
            mov  bh, cursor.X
            add  bh, -1
            mov  bl, cursor.Y
            INVOKE IsChess, bh, bl              ;判斷要跳的方向是否有棋子
            cmp  al, 0
            jz   invalidmove                    ;要跳的方向無棋子
            mov  bh, chessx
            add  bh, 2
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            mov  isjmp, 1                       ;設定 isjmp = 1
            INVOKE print
            jmp  jumpagain

        jumprightdown:
            mov  bh, cursor.X
            add  bh, -1
            mov  bl, cursor.Y
            add  bl, 1
            INVOKE IsChess, bh, bl              ;判斷要跳的方向是否有棋子
            cmp  al, 0
            jz   invalidmove                    ;要跳的方向無棋子
            mov  bh, chessx
            add  bh, 2
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            mov  bl, chessy
            add  bl, -2
            mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
            mov  isjmp, 1                       ;設定 isjmp = 1
            INVOKE print
            jmp  jumpagain
```

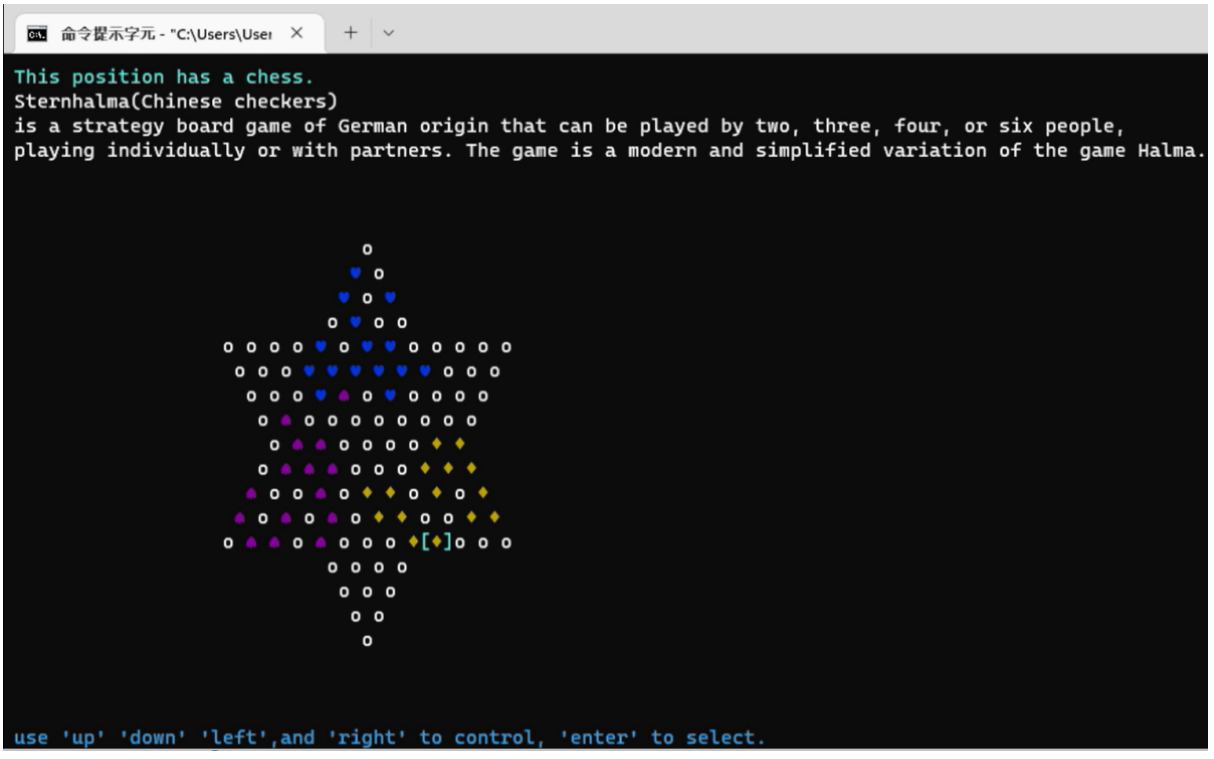
```
        xromain2:
            cmp  bl, 2
            jz   jumprightup                    ;Y座標差 = 2 (向右上跳)
            cmp  bl, -2
            jz   jumpleftdown                   ;Y座標差 = -2 (向左下跳)
            cmp  bl, 0
            jz   jumpend                        ;Y座標差 = 0 (跳躍結束)
            jmp  invalidmove                    ;Y座標差超出範圍(非法移動)

        jumprightup:
            mov  bh, cursor.X
            mov  bl, cursor.Y
            add  bl, -1
            INVOKE IsChess, bh, bl              ;判斷要跳的方向是否有棋子
            cmp  al, 0
            jz   invalidmove                    ;要跳的方向無棋子
            mov  bh, chessx
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            mov  bl, chessy
            add  bl, 2
            mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
            mov  isjmp, 1                       ;設定 isjmp = 1
            INVOKE print
            jmp  jumpagain

        jumpleftdown:
            mov  bh, cursor.X
            mov  bl, cursor.Y
            add  bl, 1
            INVOKE IsChess, bh, bl              ;判斷要跳的方向是否有棋子
            cmp  al, 0
            jz   invalidmove                    ;要跳的方向無棋子
            mov  bh, chessx
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            mov  bl, chessy
            add  bl, -2
            mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
            mov  isjmp, 1                       ;設定 isjmp = 1
            INVOKE print
            jmp  jumpagain

        xminustwo:
            cmp  bl, 2                          ;Y座標差 = 2 (向左上跳)
            jz   jumpleftup
            cmp  bl, 0                          ;Y座標差 = 0 (向左跳)
            jz   jumpleft
            jmp  invalidmove                    ;Y座標差超出範圍(非法移動)

        jumpleftup:
            mov  bh, cursor.X
            add  bh, 1
            mov  bl, cursor.Y
            add  bl, -1
            INVOKE IsChess, bh, bl              ;判斷要跳的方向是否有棋子
            cmp  al, 0
            jz   invalidmove                    ;要跳的方向無棋子
            mov  bh, chessx
            add  bh, -2
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            mov  bl, chessy
            add  bl, 2
            mov  (COOR PTR [esi]).Y, bl         ;改變旗子Y座標
            mov  isjmp, 1                       ;設定 isjmp = 1
            INVOKE print
            jmp  jumpagain

        jumpleft:
            mov  bh, cursor.X
            add  bh, 1
            mov  bl, cursor.Y
            INVOKE IsChess, bh, bl              ;判斷要跳的方向是否有棋子
            cmp  al, 0
            jz   invalidmove                    ;要跳的方向無棋子
            mov  bh, chessx
            add  bh, -2
            mov  (COOR PTR [esi]).X, bh         ;改變旗子X座標
            mov  bl, chessy
            mov  isjmp, 1                       ;設定 isjmp = 1
            INVOKE print
            jmp  jumpagain
```

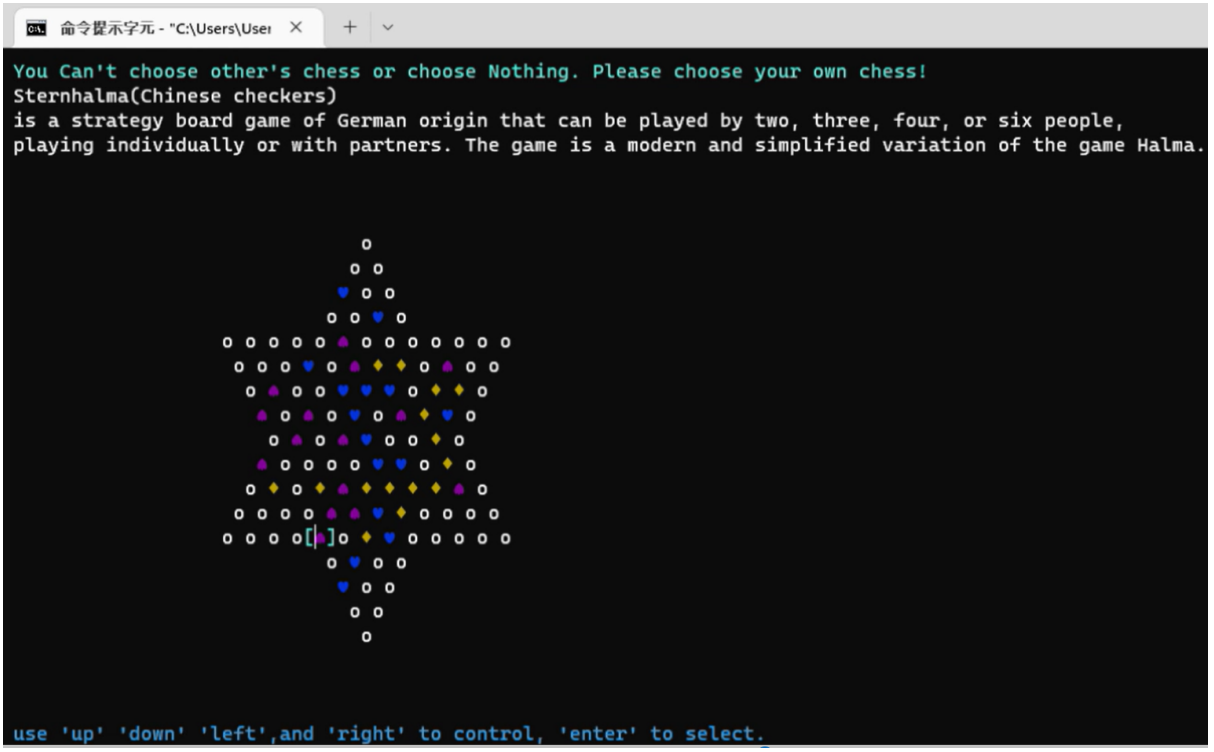
```
        ;跳棋判斷決定是否要繼續跳
        jumpagain:
            mov  dh, 0
            mov  dl, 0
            call Gotoxy
            mov  edx, OFFSET jumporstop
            call WriteString
            jmp  moveagain

        ;非法移動
        invalidmove:
            mov  dh, 0
            mov  dl, 0
            call Gotoxy
            mov  edx, OFFSET cantjump
            call WriteString
            jmp  moveagain

        ;結束移動
        jumpend:
            mov  ax, 1
            pop  edx
            pop  ebx
            ret

    movechess ENDP
```

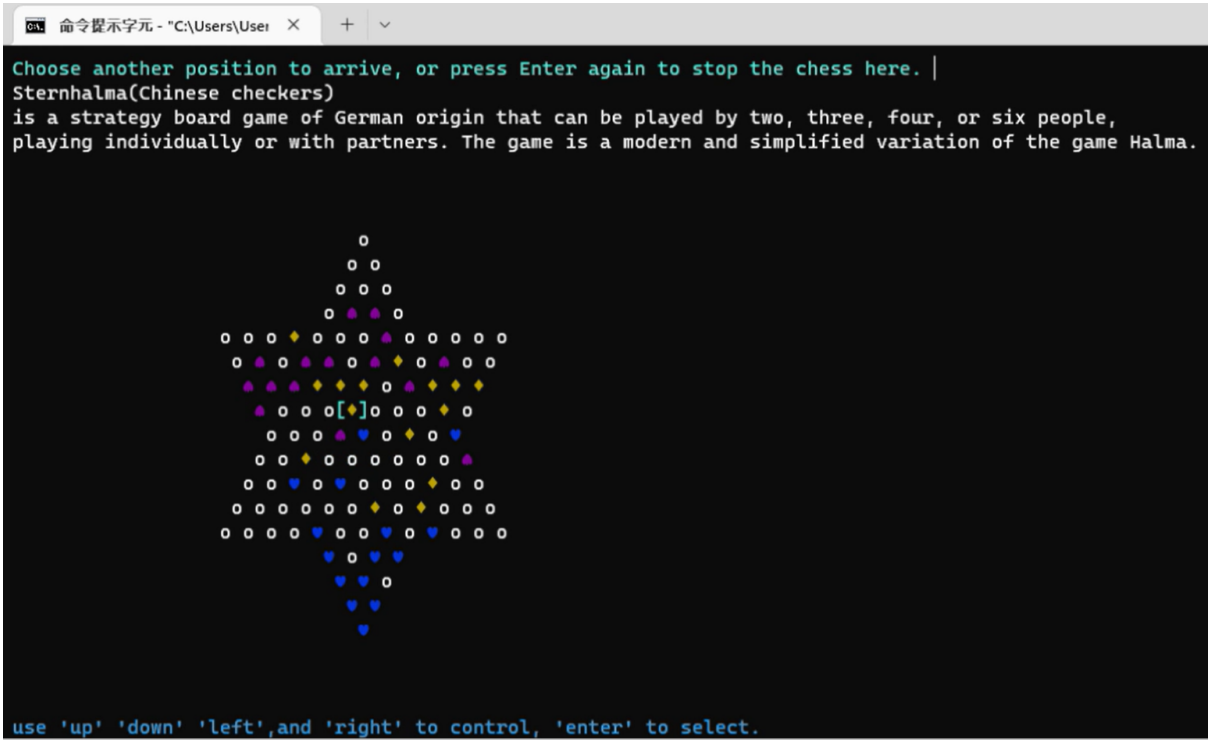
```
;-----------------------------------------------Choose-----------------------------------------------
Choose PROC
        push eax;
        push ecx;
        push ebx;
        push edx;
        push edi;
    stage_move:
        INVOKE Movecursor                   ;移动光标
        INVOKE print
        INVOKE Boundary, cursor.X, cursor.Y ;确定光标有没有在棋盘内
        cmp ax, 1
        jne stage_move                      ;没有的话回到移动光标的状态
        INVOKE IsChess, cursor.X, cursor.Y  ;确定选到的是不是正确的棋子(是不是和control相同)
        cmp al, control
        jne invalidchoose                   ;选错的话回到移动光标的状态
        mov bl, al
        mov ecx, 15                         ;count=15
        mov ah, cursor.X                    ;把光标的XY储存ax
        mov al, cursor.Y
        cmp bl, 1                           ;是蓝棋
        je chessB
        cmp bl, 2                           ;是红旗棋
        ja chessM
        cmp bl, 3                           ;是咖啡棋
        je chessBr

    chessB:
        mov edi, OFFSET B                   ;edi指到B(蓝棋)的起始位置
        jmp stage_return

    chessM:
        mov edi, OFFSET M                  ;edi指到M(红旗棋)的起始位置
        jmp stage_return

    chessBr:
        mov edi, OFFSET Br                  ;edi指到Br(咖啡棋)的起始位置
        jmp stage_return

    stage_return:
        cmp ah, (COOR PTR [edi]).X
        jnz reloop                          ;X座标不符
        cmp al, (COOR PTR [edi]).Y
        jnz reloop                          ;Y座标不符
        jmp stage_find                      ;找到相符位置的棋子

    reloop:
        add edi, TYPE COOR                  ;判断遍历的是否为下一些棋子
        loop stage_return

    stage_find:
        mov esi, edi                        ;回传esi=edi
        pop eax;
        pop ecx;
        pop ebx;
        pop edx;
        pop edi;
        ret

    ;非法选取棋子
    invalidchoose:
        mov  dh, 0
        mov  dl, 0
        call Gotoxy
        mov  edx, OFFSET chooseyourschess
        call WriteString                    ;印出"You Can't choose other's chess or choose Nothing. Please choose your own chess! "讯息
        INVOKE Transfer, cursor.X, cursor.Y
        call Gotoxy
        jmp stage_move                      ;选错的话回到移动光标的状态

Choose ENDP
;-----------------------------------------------IsChess-----------------------------------------------
```

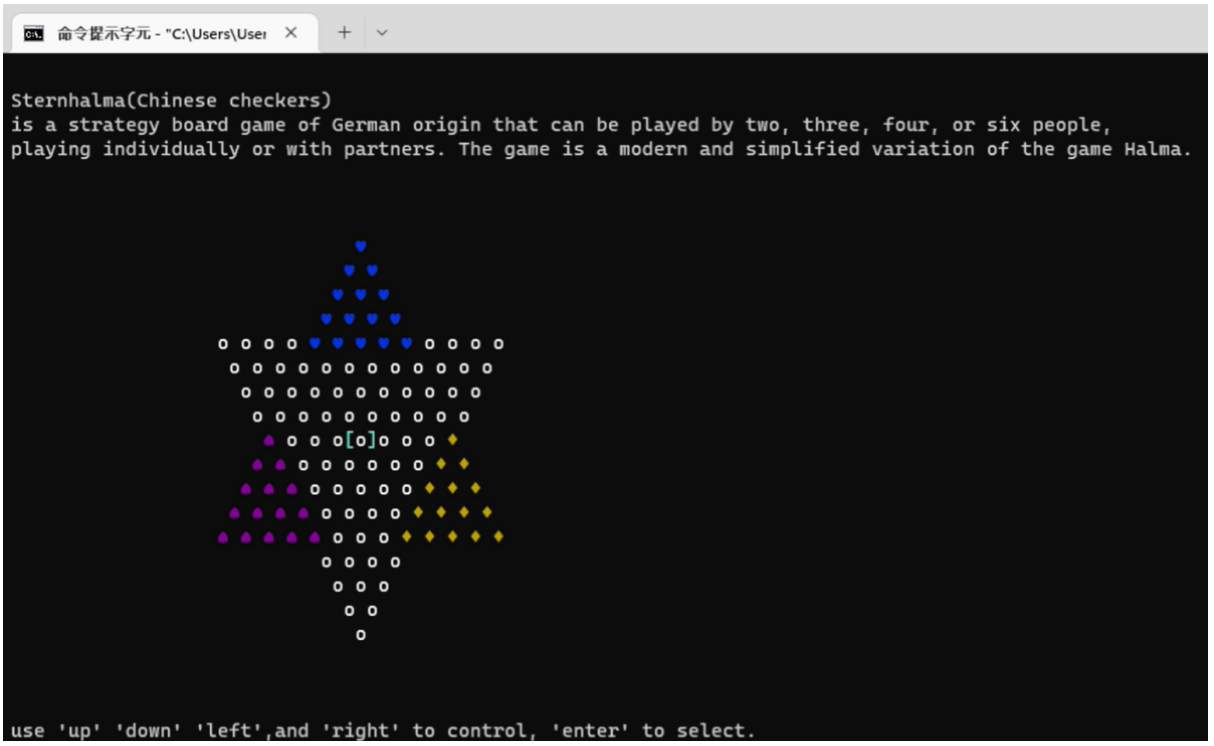
```
923  ;-----------------------------------------------IsChess-----------------------------------------------
924  IsChess PROC,
925          boolx: sbyte, booly: sbyte
926
927          ;將使用的暫存器的值存到堆疊
928          push ecx
929          push ebx
930          push edi
931
932          mov ecx, 45                         ;設定迴圈次數
933          mov edi, OFFSET B                   ;設定edi為存放的光標點位置
934      find:
935          mov bl, (COOR PTR [edi]).X
936          mov bh, (COOR PTR [edi]).Y
937
938          cmp boolx, bl
939          jnz addpointer                      ;X座標不符
940          cmp booly, bh
941          jnz addpointer                      ;Y座標不符
942          jz lookecx
943
944      addpointer:
945          add edi, TYPE COOR
946          loop find
947
948      ;找到指標位置的棋子 判斷ecx範圍 取的棋子的所有者
949      lookecx:
950          cmp ecx, 30
951          jg findr                            ;藍棋
952          cmp ecx, 15
953          jg findg                            ;紅旗棋
954          cmp ecx, 0
955          jg findy                            ;咖啡棋
956          cmp ecx, 0
957          jz nofind                           ;無棋子
958
959      ;藍棋  al = 1
960      findr:
961          mov al, 1
962          pop  edi
963          pop  ebx
964          pop  ecx
965          ret
966
967      ;紅旗棋  al = 2
968      findg:
969          mov al, 2
970          pop  edi
971          pop  ebx
972          pop  ecx
973          ret
974
975      ;咖啡棋  al = 3
976      findy:
977          mov al, 3
978          pop  edi
979          pop  ebx
980          pop  ecx
981          ret
982
983      ;無棋子   al = 0
984      nofind:
985          mov al, 0
986          pop  edi
987          pop  ebx
988          pop  ecx
989          ret
990
991  IsChess ENDP
992  END main
```

scenario of the game will indicate on the top of the scene




命令提示字元 - "C:\Users\User
You Can't choose other's chess or choose Nothing. Please choose your own chess!
Sternhalma(Chinese checkers)
is a strategy board game of German origin that can be played by two, three, four, or six people,
playing individually or with partners. The game is a modern and simplified variation of the game Halma.
use 'up' 'down' 'left',and 'right' to control, 'enter' to select.


命令提示字元 - "C:\Users\User
Choose another position to arrive, or press Enter again to stop the chess here.
Sternhalma(Chinese checkers)
is a strategy board game of German origin that can be played by two, three, four, or six people,
playing individually or with partners. The game is a modern and simplified variation of the game Halma.
use 'up' 'down' 'left',and 'right' to control, 'enter' to select.


命令提示字元 - "C:\Users\User
Sternhalma(Chinese checkers)
is a strategy board game of German origin that can be played by two, three, four, or six people,
playing individually or with partners. The game is a modern and simplified variation of the game Halma.
use 'up' 'down' 'left',and 'right' to control, 'enter' to select.