

## 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

explanation:

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

explanation:

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

explanation:

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

```
1 ;跳棋程式
2 INCLUDE Irvine32.inc
3
4 ;輸出棋盤
5 print PROTO
6
7 ;棋盤座標與小黑窗座標轉換
8 Transfer PROTO,
9     X1:sbyte,
10     Y1:sbyte
11
12 ;判斷輸入的座標是否在棋盤內
13 Boundary PROTO,
14     x:sbyte,y:sbyte
15
16 ;判斷有無玩家勝利
17 Iswin PROTO
18
19 ;判斷輸入的位置是否有棋子
20 IsChess PROTO,
21     boolx:sbyte,booly:sbyte
22
23 ;選擇棋子
24 Choose PROTO
25
26 ;移動游標
27 movecursor PROTO
28
29 ;移動棋子
30 movechess PROTO
31
32 ;棋子座標結構(帶正負的座標)
33 COOR STRUCT
34     X sbyte 0
35     Y sbyte 0
36 COOR ENDS
```

```

38 .data
39 ;15隻藍棋位置
40 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>
41 ;15隻紅紫棋位置
42 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>
43 ;15隻咖啡棋位置
44 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>
45
46 ;快贏的15隻藍棋位置
47 ;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>
48 ;快贏的15隻紅紫棋位置
49 ;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>
50 ;快贏的15隻咖啡棋位置
51 ;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>
52
53 ;棋盤樣式
54 dot byte "o"
55
56 ;游標位置
57 cursor COOR < 0, 0>
58
59 ;控制權
60 control byte 1
61
62 ;遊戲狀態
63 chos byte "choose",0
64 unchos byte "unlock",0
65 En byte "end",0
66 bluewin byte "The winner is blue!!!",0
67 magentawin byte "The winner is magenta!!!",0
68 brownwin byte "The winner is yellow!!!",0
69 jumporstop byte "Choose another position to arrive, or press Enter again to stop the chess here. ",0
70 havechess byte "This position has a chess. ",0
71 cantjump byte "You can't jump to the position",0
72 chooseyourschess byte "You Can't choose other's chess or choose Nothing. Please choose your own chess! ",0
73
74 introduction byte "Sternhalma(Chinese checkers)",0
75 introduction2 byte "is a strategy board game of German origin that can be played by two, three, four, or six people.",0
76 introduction3 byte "playing individually or with partners. The game is a modern and simplified variation of the game Halma.",0
77 way_to_play byte "use 'up' 'down' 'left',and 'right' to control, 'enter' to select. ",0

```

```

79 .code
80 main PROC
81
82     INVOKE print                ;印出初始棋盤
83
84     gameconti:
85
86         INVOKE Iswin            ;判斷有無玩家勝利
87         cmp ax,1
88         jz somebodywin         ;有玩家勝利
89         INVOKE Choose          ;選擇棋子
90         mov edx, OFFSET chos
91         call WriteString        ;輸出"choose"狀態
92         INVOKE movechess       ;移動棋子
93         cmp ax, 0
94         jz unchoose           ;重新選擇棋子
95         INVOKE print           ;輸出棋盤
96         mov edx, OFFSET En
97         call WriteString       ;輸出"end"狀態
98
99         add control, 1         ;control = (control + 1) % 3 + 1
100        cmp control, 3
101        jng gameconti
102        add control, -3
103        mov al,control
104        jmp gameconti          ;遊戲繼續
105
106    unchoose:                   ;重新選擇棋子
107        INVOKE print           ;輸出棋盤
108        mov edx, OFFSET unchos
109        call WriteString        ;輸出"unlock"狀態
110        jmp gameconti
111
112    somebodywin:                ;有玩家勝利
113        cmp bx,1
114        jz wblue
115        cmp bx,2
116        jz wmagenta
117        cmp bx,3
118        jz wbrown
119
120    wblue:                      ;藍棋勝利
121        mov dh, 0
122        mov dl, 0
123        call Gotoxy
124        mov edx, OFFSET bluwin
125        call WriteString        ;印出"The winner is blue!!!"狀態
126        jmp endgame
127
128    wmagenta:                   ;紅紫棋勝利
129        mov dh, 0
130        mov dl, 0
131        call Gotoxy
132        mov edx, OFFSET magentawin
133        call WriteString        ;印出"The winner is magenta!!!"狀態
134        jmp endgame
135
136    wbrown:                     ;咖啡棋勝利
137        mov dh, 0
138        mov dl, 0
139        call Gotoxy
140        mov edx, OFFSET brownwin
141        call WriteString        ;印出"The winner is brown!!!"狀態
142        jmp endgame
143
144    endgame:                    ;遊戲結束
145        call readchar
146        exit
147
148 main ENDP

```

```

150 print PROC
151
152         call clrscr                                ;清空螢幕
153
154         push eax                                    ;將會用到的暫存器內容放入堆疊
155         push ebx
156         push ecx
157         push esi
158
159         ;---繪製大棋盤白點---
160         ;雙層迴圈 -8 <= i,j <= 8
161         ;透過 Boundary 以及透過 Transfer 轉換, 輸出棋盤上的點出棋盤上的點
162         mov ecx, 17
163         printintro:
164
165         mov al, 15
166         call SetTextColor
167         mov dh, 1
168         mov dl, 0
169         call Gotoxy
170         mov edx, OFFSET introduction
171         call WriteString
172         mov dh, 2
173         mov dl, 0
174         call Gotoxy
175         mov edx, OFFSET introduction2
176         call WriteString
177         mov dh, 3
178         mov dl, 0
179         call Gotoxy
180         mov edx, OFFSET introduction3
181         call WriteString
182         mov al, 3
183         call SetTextColor
184         mov dh, 27
185         mov dl, 0
186         call Gotoxy
187         mov edx, OFFSET may_to_play
188         call WriteString
189         outter:
190         mov bh, cl
191         sub bh, 9
192
193         push ecx
194
195         mov ecx, 17
196         inner:
197         mov bl, cl
198         sub bl, 9
199         INVOKE Boundary, bh, bl
200         cmp ax, 0
201         jz outofbound                                ;點出界 --> 不印出
202
203         INVOKE Transfer, bh, bl
204         call Gotoxy                                    ;游標位置已經由 Transfer 算好, 並放入適當的暫存器 (dl,dh)
205         mov eax, white + ( black*16 )                ;設定前景為白色, 背景為黑色
206         call SetTextColor
207
208         mov al, dot
209         call Writechar                                ;印出點點
210
211         outofbound:
212
213         loop inner
214
215         pop ecx
216
217         loop outter

```

```

219 ;---大棋盤紅點---
220 ;按照矩陣的內容, 透過 Transfer 轉換後輸出
221 mov ecx, 15
222 mov esi, OFFSET B
223
224 printblue:
225 mov bh, (COORD PTR [esi]).X
226 mov bl, (COORD PTR [esi]).Y
227 INVOKE Transfer, bh, bl
228 call Gotoxy
229 mov eax, 1 + ( black*16 ) ;設定前景為藍色, 背景為黑色
230 call SetTextColor
231
232 mov al, 3h
233 call Writechar ;印出藍心
234
235 add esi, TYPE COOR
236 loop printblue
237
238 ;---大棋盤紅紫點---
239 ;按照矩陣的內容, 透過 Transfer 轉換後輸出
240 mov ecx, 15
241 mov esi, OFFSET M
242
243
244 printmagenta:
245 mov bh, (COORD PTR [esi]).X
246 mov bl, (COORD PTR [esi]).Y
247 INVOKE Transfer, bh, bl
248 call Gotoxy
249 mov eax, 5 + ( black*16 ) ;設定前景為紅紫色, 背景為黑色
250 call SetTextColor
251
252 mov al, 6h
253 call Writechar ;印出黑桃
254
255 add esi, TYPE COOR
256 loop printmagenta
257
258 ;---大棋盤咖啡點---
259 ;按照矩陣的內容, 透過 Transfer 轉換後輸出
260 mov ecx, 15
261 mov esi, OFFSET Br
262
263
264 printbrown:
265 mov bh, (COORD PTR [esi]).X
266 mov bl, (COORD PTR [esi]).Y
267 INVOKE Transfer, bh, bl
268 call Gotoxy
269 mov eax, 6 + ( black*16 ) ;設定前景為咖啡色, 背景為黑色
270 call SetTextColor
271
272 mov al, 4h
273 call Writechar ;印出菱形
274 add esi, TYPE COOR
275 loop printbrown

```

```

276 ;---大棋盤游標---
277 mov bh, cursor.X
278 mov bl, cursor.Y
279 INVOKE Transfer, bh, bl
280
281 ;繪製 "[" , 淡青綠色
282 sub dl, 1
283 call Gotoxy
284 mov eax, 11 + ( black*16 )
285 call SetTextColor
286 mov al, "["
287 call Writechar
288
289 ;繪製 "]" , 淡青綠色
290 add dl, 2
291 call Gotoxy
292 mov eax, 11 + ( black*16 )
293 call SetTextColor
294 mov al, "]"
295 call Writechar
296
297 ;從堆疊取回暫存器內容
298 pop esi
299 pop ecx
300 pop ebx
301 pop eax
302
303 ret
304
305 print ENDP

```

```
307 ;-----Transfer-----
```

```
308 Transfer PROC,  
309     X1:sbyte, Y1:sbyte  
310  
311     ;將會用到的暫存器內容放入堆疊  
312     push eax  
313     push ebx  
314  
315     mov al, X1;  
316     mov bl, Y1;  
317  
318     ;2*x1 + y1 + 13 =x2  
319  
320     add al, al  
321     add al, bl  
322     add al, 30  
323     mov dl, al  
324  
325     ;-y1 + 9 = y2  
326     neg bl  
327     add bl, 15  
328     mov dh, bl  
329  
330     ;從堆疊取回暫存器內容  
331     pop ebx  
332     pop eax  
333  
334     ret
```

```
335 Transfer ENDP
```

```
336 ;-----Boundary-----
```

```
337  
338 Boundary PROC, ux:SBYTE, uy:SBYTE  
339     push ebx  
340     test_uptri:  
341  
342     ;判斷是否在上三角裡。只要有一項不符合。  
343     ;就直接跳去判斷是不是下三角里  
344     cmp uy, -4  
345     jl test_downtri ;y<-4  
346     cmp ux, -4  
347     jl test_downtri ;x<-4  
348     mov bl, ux  
349     add bl, uy  
350     cmp bl, 4  
351     jg test_downtri ;x+y>4  
352     jmp Istrue  
353  
354     ;三項都符合。所以在上三角裡  
355     ;就直接跳去Istrue。回傳true(ax=1)。結束程式  
356  
357     test_downtri:  
358     ;判斷是否在下三角裡。只要有一項不符合。  
359     ;就直接跳去Isfalse。回傳false(ax=0)。結束程式  
360     cmp uy, 4  
361     jg Isfalse ;y > 4  
362     cmp ux, 4  
363     jg Isfalse ;x > 4  
364     mov bl, ux  
365     add bl, uy  
366     cmp bl, -4  
367     jl Isfalse ;x+y<=-4  
368  
369     ;三項都符合。所以在下三角裡就  
370     ;到Istrue。回傳true(ax=1)。結束程式  
371  
372     Istrue:  
373     mov ax, 1  
374     jmp existBoundary  
375     Isfalse:  
376     mov ax, 0  
377     existBoundary:  
378     pop ebx  
379     ret  
380  
381 Boundary ENDP
```



```

383 Iswin PROC
384     push edi
385     push ecx
386     mov ecx, 15
387     mov edi, 0
388     checkR:
389         cmp (COORD PTR B[edi]).Y, -4           ;B的每個棋子的Y都必須小於等於-4
390         jg checkM                               ;只要有一顆沒有，B就不可能贏。就直接跳去check M
391         cmp (COORD PTR B[edi]).X, 0           ;B的每個棋子的X都必須大於等於0
392         jl checkM                               ;只要有一顆沒有，B就不可能贏。就直接跳去check M
393         cmp (COORD PTR B[edi]).X, 4           ;B的每個棋子的X都必須小於等於4
394         jg checkM                               ;只要有一顆沒有，B就不可能贏。就直接跳去check M
395         add edi, TYPE COOR
396         loop checkR
397         mov bx, 1
398         jmp Win                                ;B每顆棋子的Y都小於等於-4，則B贏了
399
400     checkM:
401         mov ecx, 15
402         mov edi, 0
403     Mloop:
404         mov bl, (COORD PTR M[edi]).X           ;M的每個棋子都必須x+y>=4
405         add bl, (COORD PTR M[edi]).Y
406         cmp bl, 4
407         jl checkBr                             ;只要有一顆沒有，M就不可能贏。就直接跳去check Br
408         cmp (COORD PTR M[edi]).Y, 0           ;M的每個棋子的Y都必須大於等於0
409         jl checkBr                             ;只要有一顆沒有，M就不可能贏。就直接跳去check Br
410         cmp (COORD PTR M[edi]).Y, 4           ;M的每個棋子的Y都必須小於等於4
411         jg checkBr                             ;只要有一顆沒有，M就不可能贏。就直接跳去check Br
412         add edi, TYPE COOR
413         loop Mloop
414         mov bx, 2
415         jmp Win                                ;M的每個棋子都x+y>=4，則M贏了
416
417     checkBr:
418         mov ecx, 15
419         mov edi, 0
420     Brloop:
421         cmp (COORD PTR Br[edi]).X, -4         ;Br的每個棋子的X都必須小於等於-4
422         jg Conti                               ;只要有一顆沒有，Br就不可能贏。即B,M,Br都沒有人贏
423         cmp (COORD PTR Br[edi]).Y, 0         ;Br的每個棋子的Y都必須大於等於0
424         jl Conti                               ;只要有一顆沒有，Br就不可能贏。即B,M,Br都沒有人贏
425         cmp (COORD PTR Br[edi]).Y, 4         ;Br的每個棋子的Y都必須小於等於4
426         jg Conti                               ;只要有一顆沒有，Br就不可能贏。即B,M,Br都沒有人贏
427         add edi, TYPE COOR
428         loop Brloop
429         mov bx, 3
430         jmp Win                                ;Br每顆棋子的X都小於等於-4，則Br贏了
431
432     Win:                                       ;有玩家勝利 ax = 1
433         mov ax, 1
434         jmp existIswin
435
436     Conti:                                   ;無玩家勝利 ax = 0
437         mov ax, 0;
438
439     existIswin:
440         pop ecx
441         pop edi
442         ret
443
444 Iswin ENDP

```

```

445
446 movecursor PROC
447
448     push eax
449     push ebx
450
451     ;等待輸入上、下、左、右、enter
452     WaitInput:
453         call readchar
454         cmp eax, 4808h           ;上
455         jz UP
456         cmp eax, 5008h           ;下
457         jz DOWN
458         cmp eax, 4008h           ;左
459         jz LEFT
460         cmp eax, 4098h           ;右
461         jz RIGHT
462         cmp eax, 1C00h           ;enter
463         jz OUTFUN
464         jmp WaitInput
465
466     ;上 -> Y座標+1 並判斷位置是否超出邊界
467     UP:
468         mov bh, cursor.X
469         mov bl, cursor.Y
470         add bl, 1
471         INVOKE Boundary, bh, bl
472         cmp ax, 0
473         jz WaitInput
474         mov cursor.X, bh
475         mov cursor.Y, bl
476         INVOKE print
477         jmp WaitInput
478
479     ;下 -> Y座標-1 並判斷位置是否超出邊界
480     DOWN:
481         mov bh, cursor.X
482         mov bl, cursor.Y
483         add bl, -1
484         INVOKE Boundary, bh, bl
485         cmp ax, 0
486         jz WaitInput
487         mov cursor.X, bh
488         mov cursor.Y, bl
489         INVOKE print
490         jmp WaitInput
491
492     ;左 -> X座標-1 並判斷位置是否超出邊界
493     LEFT:
494         mov bh, cursor.X
495         mov bl, cursor.Y
496         add bh, -1
497         INVOKE Boundary, bh, bl
498         cmp ax, 0
499         jz WaitInput
500         mov cursor.X, bh
501         mov cursor.Y, bl
502         INVOKE print
503         jmp WaitInput
504
505     ;右 -> X座標+1 並判斷位置是否超出邊界
506     RIGHT:
507         mov bh, cursor.X
508         mov bl, cursor.Y
509         add bh, 1
510         INVOKE Boundary, bh, bl
511         cmp ax, 0
512         jz WaitInput
513         mov cursor.X, bh
514         mov cursor.Y, bl
515         INVOKE print
516         jmp WaitInput
517
518     ;enter 跳出函式
519     OUTFUN:
520         pop ebx
521         pop eax
522         ret
523
524 movecursor ENDP

```

```

525 ;-----movechess-----
526 movechess PROC
527
528 ;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
529 LOCAL isjmp: byte, chessx: sbyte, chessy: sbyte
530 push ebx
531 push edx
532
533 ;isjmp預設為0
534 mov isjmp, 0
535
536 moveagain:
537 ;移動游標
538 INVOKE movecursor
539
540 ;將棋子原始位置寫入 (bh, bl)
541 mov bh, (COORD PTR [esi]).X
542 mov chessx, bh
543 mov bl, (COORD PTR [esi]).Y
544 mov chessy, bl
545
546 ;檢查游標位置是否有棋子
547 INVOKE IsChess, cursor.X, cursor.Y
548 cmp al, 0
549 jz startmove ;該位置無棋子
550
551 ;如果(bh, bl) == (cursor.X, cursor.Y) => 處理棋結束 或者 取消選取
552 cmp bh, cursor.X
553 jnz haschess
554 cmp bl, cursor.Y
555 jnz haschess
556 jmp startmove
557
558 haschess:
559 mov dh, 0
560 mov dl, 0
561 call Gotoxy
562 mov edx, OFFSET havechess ;印出"This position has a chess."狀態
563 call WriteString
564 jnz moveagain
565
566 startmove:
567 ;(bh, bl) 記錄後來位置與原始位置的差
568 mov bh, cursor.X
569 mov bl, cursor.Y
570 sub bh, chessx
571 sub bl, chessy
572
573 ;跳過的棋子不用判斷移動
574 cmp isjmp, 0
575 jnz jump ;處理"跳"棋
576
577 cmp bh, 1
578 jz xplusone ;X座標差 = +1(走、跳)
579 cmp bh, 0
580 jz xremain1 ;X座標差 = 0(走、跳、取消選取)
581 cmp bh, -1
582 jz xminusone ;X座標差 = -1(走、跳)
583 cmp bh, 2
584 jz xplustwo ;X座標差 = +2(跳)
585 cmp bh, -2
586 jz xminustwo ;X座標差 = -2(跳)
587 jmp invalidmove ;X座標差超出範圍(非法移動)
588
589 ;X座標差 = +1(走、跳)
590 xplusone:
591 cmp bl, 0
592 jz moveright; ;Y座標差 = 0(向右走)
593 cmp bl, -1
594 jz moverightdown ;Y座標差 = -1(向右下走)
595 jmp invalidmove ;Y座標差超出範圍(非法移動)
596
597

```

```

598 ;同右起
599 moveright:
600     mov bh, chessx
601     add bh, 1
602     mov (COORD 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 ;向右上走
609 moverightdown:
610     mov bh, chessx
611     add bh, 1
612     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
613     mov bl, chessy
614     add bl, -1
615     mov (COORD PTR [esi]).Y, bl ;改變旗子Y座標
616     INVOKE print
617     mov ax, 1
618     pop ebx
619     ret
620
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 (COORD PTR [esi]).X, bh ;改變旗子X座標
645     mov bl, chessy
646     add bl, 1
647     mov (COORD 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 (COORD PTR [esi]).X, bh ;改變旗子X座標
657     mov bl, chessy
658     sub bl, 1
659     mov (COORD 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)

```

```

774
775 ;向左走
776 moveleft:
777     mov bh, chessx
778     add bh, -1
779     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
780     INVOKE print
781     mov ax, 1
782     pop ebx
783     ret
784
785 ;向左上走
786 moveleftup:
787     mov bh, chessx
788     add bh, -1
789     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
790     mov bl, chessy
791     add bl, 1
792     mov (COORD PTR [esi]).Y, bl ;改變旗子Y座標
793     INVOKE print
794     mov ax, 1
795     pop ebx
796     ret
797
798 ;這裡處理"跳"棋
799 jump:
800     cmp bh, 2
801     jz xplustwo ;X座標差 = +2(跳)
802     cmp bh, 0
803     jz xremain2 ;X座標差 = 0(跳)
804     cmp bh, -2
805     jz xminustwo ;X座標差 = -2(跳)
806     jmp invalidmove ;X座標差超出範圍(非法移動)
807
808 xplustwo:
809     cmp bl, 0
810     jz jumpright ;Y座標差 = 0(向右跳)
811     cmp bl, -2
812     jz jumprightdown ;Y座標差 = -2(向右下跳)
813     jmp invalidmove ;Y座標差超出範圍(非法移動)
814
815 jumpright:
816     mov bh, cursor.X
817     add bh, -1
818     mov bl, cursor.Y
819     INVOKE IsChess, bh, bl ;判斷要跳的方向是否有棋子
820     cmp al, 0
821     jz invalidmove ;要跳的方向無棋子
822     mov bh, chessx
823     add bh, 2
824     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
825     mov isjmp, 1 ;設定 isjmp = 1
826     INVOKE print
827     jmp jumpagain
828
829 jumprightdown:
830     mov bh, cursor.X
831     add bh, -1
832     mov bl, cursor.Y
833     add bl, 1
834     INVOKE IsChess, bh, bl ;判斷要跳的方向是否有棋子
835     cmp al, 0
836     jz invalidmove ;要跳的方向無棋子
837     mov bh, chessx
838     add bh, 2
839     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
840     mov bl, chessy
841     add bl, -2
842     mov (COORD PTR [esi]).Y, bl ;改變旗子Y座標
843     mov isjmp, 1 ;設定 isjmp = 1
844     INVOKE print
845     jmp jumpagain

```

```

745 xremain2:
746     cmp bl, 2
747     jz  jumprightup           ;V座標差 = 2 (向右上跳)
748     cmp bl, -2
749     jz  jumpleftdown         ;V座標差 = -2 (向左下跳)
750     cmp bl, 0
751     jz  jumpend              ;V座標差 = 0 (跳到此處)
752     jmp invalidmove          ;V座標差超出範圍(非法移動)
753
754 jumprightup:
755     mov bh, cursor.X
756     mov bl, cursor.Y
757     add bl, -1
758     INVOKE IsChess, bh, bl    ;判斷要跳的方向是否有棋子
759     cmp al, 0
760     jz  invalidmove          ;要跳的方向無棋子
761     mov bh, chessx
762     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
763     mov bl, chessy
764     add bl, 2
765     mov (COORD PTR [esi]).Y, bl ;改變旗子Y座標
766     mov isjep, 1              ;設定 isjep = 1
767     INVOKE print
768     jmp jumpagain
769
770 jumpleftdown:
771     mov bh, cursor.X
772     mov bl, cursor.Y
773     add bl, 1
774     INVOKE IsChess, bh, bl    ;判斷要跳的方向是否有棋子
775     cmp al, 0
776     jz  invalidmove          ;要跳的方向無棋子
777     mov bh, chessx
778     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
779     mov bl, chessy
780     add bl, -2
781     mov (COORD PTR [esi]).Y, bl ;改變旗子Y座標
782     mov isjep, 1              ;設定 isjep = 1
783     INVOKE print
784     jmp jumpagain
785
786 xminustwo:
787     cmp bl, 2
788     jz  jumpleftup           ;V座標差 = 2 (向左上跳)
789     cmp bl, 0
790     jz  jumpleft             ;V座標差 = 0 (向左跳)
791     jmp invalidmove          ;V座標差超出範圍(非法移動)
792
793 jumpleftup:
794     mov bh, cursor.X
795     add bh, 1
796     mov bl, cursor.Y
797     add bl, -1
798     INVOKE IsChess, bh, bl    ;判斷要跳的方向是否有棋子
799     cmp al, 0
800     jz  invalidmove          ;要跳的方向無棋子
801     mov bh, chessx
802     add bh, -2
803     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
804     mov bl, chessy
805     add bl, 2
806     mov (COORD PTR [esi]).Y, bl ;改變旗子Y座標
807     mov isjep, 1              ;設定 isjep = 1
808     INVOKE print
809     jmp jumpagain
810
811 jumpleft:
812     mov bh, cursor.X
813     add bh, 1
814     mov bl, cursor.Y
815     INVOKE IsChess, bh, bl    ;判斷要跳的方向是否有棋子
816     cmp al, 0
817     jz  invalidmove          ;要跳的方向無棋子
818     mov bh, chessx
819     add bh, -2
820     mov (COORD PTR [esi]).X, bh ;改變旗子X座標
821     mov bl, chessy
822     mov isjep, 1              ;設定 isjep = 1
823     INVOKE print
824     jmp jumpagain

```

```

825 ;跳躍可以決定是否要繼續跳
826 jumpagain:
827     mov dh, 0
828     mov dl, 0
829     call Gotoxy
830     mov edx, OFFSET jumprstop
831     call WriteString
832     jmp moveagain
833
834 ;非法移動
835 invalidmove:
836     mov dh, 0
837     mov dl, 0
838     call Gotoxy
839     mov edx, OFFSET cantjump
840     call WriteString
841     jmp moveagain
842
843 ;結束移動
844 jumpend:
845     mov ax, 1
846     pop edx
847     pop ebx
848     ret
849
850 movechess ENDP

```

```

852 ;----- Choose -----
853 Choose PROC
854     push eax;
855     push ecx;
856     push ebx;
857     push edx;
858     push edi;
859     stage_move:
860         INVOKE Movecursor           ;移动游标
861         INVOKE print
862         INVOKE Boundary, cursor.X, cursor.Y ;验证游标有没有在棋盘内
863         cmp ax, 1
864         jne stage_move             ;没有的话回到移动游标的状态
865         INVOKE IsChess, cursor.X, cursor.Y ;验证遇到的是不是正好的棋子(是不是和control相同)
866         cmp al, control
867         jne invalidchoose         ;遇到的话回到移动游标的状态
868         mov bl, al
869         mov ecx, 15                ;count=15
870         mov ah, cursor.X          ;把游标的XY传进ax
871         mov al, cursor.Y
872         cmp bl, 1                 ;是蓝棋
873         je chessB
874         cmp bl, 2                 ;是红紫棋
875         je chessM
876         cmp bl, 3                 ;是咖啡棋
877         je chessBr
878
879     chessB:
880         mov edi, OFFSET B         ;edi指向B(蓝棋)的起始位置
881         jmp stage_return
882
883     chessM:
884         mov edi, OFFSET M         ;edi指向M(红紫棋)的起始位置
885         jmp stage_return
886
887     chessBr:
888         mov edi, OFFSET Br        ;edi指向Br(咖啡棋)的起始位置
889         jmp stage_return
890
891     stage_return:
892         cmp ah, (COORD PTR [edi]).X
893         jnz reloop                ;X坐标不符
894         cmp al, (COORD PTR [edi]).Y
895         jnz reloop                ;Y坐标不符
896         jmp stage_find            ;找到相符位置的棋子
897
898     reloop:
899         add edi, TYPE COORD        ;判断遇到的是否为下一枚棋子
900         loop stage_return
901
902     stage_find:
903         mov esi, edi              ;记得esi=edi
904         pop eax;
905         pop ecx;
906         pop ebx;
907         pop edx;
908         pop edi;
909         ret
910
911 ;非法输入棋子
912 invalidchoose:
913     mov dh, 8
914     mov dl, 8
915     call Gotoxy
916     mov edx, OFFSET chooseyourchess
917     call WriteString              ;印出"You Can't choose other's chess or choose Nothing. Please choose your own chess!"状态
918     INVOKE Transfer, cursor.X, cursor.Y
919     call Gotoxy
920     jmp stage_move               ;遇到的话回到移动游标的状态
921
922 Choose ENDP
923 ;----- ToChess -----

```

```

923 ;-----IsChess
924 IsChess PROC,
925     boolx: sbyte, booly: sbyte
926
927     ;將使用的暫存器的值存到堆棧
928     push ecx
929     push ebx
930     push edi
931
932     mov ecx, 45 ;設定回圈次數
933     mov edi, OFFSET B ;設定edi指到的地址的軸位置
934     find:
935         mov bl, (COORD PTR [edi]).X
936         mov bh, (COORD 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 COORD
946         loop find
947
948     ;找到對應位置的棋子 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 ;紅棋
954         cmp ecx, 8
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

The game board is a large star shape formed by white circles. It has 7 points. Blue hearts are placed on the top-left arm, and purple triangles are placed on the bottom-left arm. A green cursor is positioned at the bottom-left point of the star.

use 'up' 'down' 'left'.and 'right' to control. 'enter' to select.

```

This position has a 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.

      o
     ♥ o
    ♥ o ♥
   o ♥ o o
  o o o o ♥ o ♥ ♥ o o o o o
 o o o ♥ ♥ ♥ ♥ ♥ o o o
  o o o ♥ ♠ o ♥ o o o o
   o ♠ o o o o o o o
    o ♠ ♠ o o o o ♦ ♦
   o ♠ ♠ ♠ o o o ♦ ♦ ♦
  ♠ o o ♠ o ♦ o ♦ o ♦
 ♠ o ♠ o o o ♦ ♦ o o ♦
o ♠ ♠ ♠ o o o ♦ [♥] o o
      o o o o
      o o o
      o o
      o

use 'up' 'down' 'left' and 'right' to control. 'enter' to select.

```

```

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.

      0
     0 0
    ♠ 0 0
   0 0 ♠ 0
  0 0 0 0 ♠ 0 0 0 0 0 0
 0 0 0 ♠ 0 ♠ ♠ 0 0 ♠ 0 0
 0 ♠ 0 0 ♠ ♠ ♠ 0 ♠ ♠ 0
  ♠ 0 ♠ 0 ♠ 0 ♠ ♠ ♠ ♠ 0
    0 ♠ 0 ♠ ♠ ♠ 0 ♠ ♠ 0
  ♠ 0 0 0 0 ♠ ♠ ♠ ♠ 0
 0 ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ 0
 0 0 0 0 ♠ ♠ ♠ ♠ 0 0 0 0
 0 0 0 0 [ ] ♠ ♠ 0 0 0 0
      0 ♠ 0 0
     ♠ 0 0
      0 0
       0

use 'up' 'down' 'left',and 'right' to control, 'enter' to select.

```

