4 in a Row

Fonaments dels computadors - Nivell avançat

Autors:
ALBERT CAIRE RODRÍGUEZ
NIU: 1702574
MARÍA FLORIDO MUÑIZ
NIU: 1710768
EDUARDO PÉREZ MOTATO
NIU: 1709992

30 de desembre de 2023

1 Descripció

Hola amigos

2 Objectius

Que tal, aquí yo

3 Variables utilitzades

- Hola
- Que tal

4 Subrutines

4.1 Nivell basic

Uououo nivel basico

4.1.1 showCursor

```
246 showCursor:
247     push rbp
248     mov rbp, rsp
249 
250     ;Salvar registres a la pila
251     push rax
252     push rbx
253     ;Inicialització dels registres
254     mov eax, 0
255     mov ebx, 0
256 
257     ;Codi de la Pràctica
258     ;rowScreen=rowScreenIni
259     mov eax, DWORD [RowScreenIni]
260     mov DWORD [rowScreen], eax
261     ;colScreen=colScreenIni+(colCursor*4)
262     mov al, BYTE [colCursor]
263     sub eax, 'A' ;Passem de codi ASCII a nombre restant-li 'A' (=65)
264     shl eax, 2
265     add eax, DWORD [ColScreenIni]
266     mov DWORD [colScreen], eax
```

```
267
268     call gotoxy ;Sent x=[rowScreen] i y=[colScreen]
269
270     ;Recuperar registres de la pila
271     pop rbx
272     pop rax
273
274     mov rsp, rbp
275     pop rbp
276     ret
```

4.1.2 showPlayer

```
294 showPlayer:
295     push rbp
296     mov rbp, rsp
297
298     ;Salvar registres a la pila
299     push rax
300
301     ;Passar DWORD [player] a ASCII i mostrar-lo segons
302     ;[rowScreen] i [colScreen]
303     mov eax, DWORD [player]
304     add eax, 48;Passem el nombre a codi ASCII (1=49 i 2=50)
305     mov DWORD [rowScreen], 23
306     mov DWORD [colScreen], 20
307
308     ;Cridar gotoxy per posicionar cursor
309     call gotoxy
310
311     ;Emmagatzemar valor de caràcter a dil
312     mov dil, al ;al conté el valor de player+65 (player és un enter)
313
314     ;Cridar printch per mostrar caràcter
315     call printch
316
317     ;Recuperar registres de la pila
318     pop rax
319
320     mov rsp, rbp
321     pop rbp
322     ret
```

4.1.3 showBoard

```
337 showBoard:
```

```
338     push rbp
339     mov rbp, rsp
340 
341     ;Salvar registres a la pila
342     push rsi
343     push rdi
344     push rax
345 
346     ;Inicialitzar registres
347     mov eax, 0
348     mov rdi, 0
349     mov rsi, 0
350 
351     ;Fila i columna per començar des del principi
352 
353     mov DWORD [rowScreen], 8
354     buclefilas:
355        cmp DWORD [rowScreen], 20
356        jge fi_showBoard
357        add DWORD [rowScreen], 2
358 
359        mov DWORD [colScreen], 4
360     buclecolumnas:
361        cmp DWORD [colScreen], 32
362        jge buclefilas
363        add DWORD [colScreen], 4
364        mov dil, BYTE [mBoard + eax]
365        inc eax
366 
367        ;Cridar gotoxy per posicionar cursor
368        call gotoxy
369 
370        ;Cridar printch per mostrar caràcter (segons el contingut de dil)
371        call printch
372 
373        ;Reiniciem el bucle
374        jmp buclecolumnas
375 
376     fi_showBoard:
377 
378     ;Cridar showPlayer per mostrar el jugador
379     call showPlayer
380 
381     ;Recuperar registres de la pila
382     pop rax
383     pop rdi
384     pop rsi
385 
386     mov rsp, rbp
```

```
387     pop rbp
388     ret
```

4.1.4 moveCursor

```
406 moveCursor:
407     push rbp
408     mov rbp, rsp
409     call showCursor ;Mostrem el cursor on toca
410     bucle_moveCursor:
411         call getch
412         cmp BYTE [tecla], 'j'
413         je moveLeft
414         cmp BYTE [tecla], 'k'
415         je moveRight
416         cmp BYTE [tecla], '␣'
417         je fi_moveCursor
418         cmp BYTE[tecla], 27;En ASCII 27és [ESC]
419         je fi_moveCursor
420         jmp bucle_moveCursor
421 
422     moveLeft:
423         cmp BYTE [colCursor], 'A'
424         je fi_moveCursor
425         dec BYTE [colCursor]
426         jmp fi_moveCursor
427 
428     moveRight:
429         cmp BYTE [colCursor], 'G'
430         je fi_moveCursor
431         inc BYTE [colCursor]
432 
433     fi_moveCursor:
434         call showCursor
435 
436     mov rsp, rbp
437     pop rbp
438     ret
```

4.1.5 moveCursorContinuous

```
454 moveCursorContinuous:
455     push rbp
456     mov rbp, rsp
457 
458     bucle_moveCursorContinuous:
```

```
459         call moveCursor
460         cmp BYTE [tecla], '␣'
461         je fi_moveCursorContinuous
462         cmp BYTE [tecla], 27;En ASCII 27és [ESC]
463         je fi_moveCursorContinuous
464         jmp bucle_moveCursorContinuous
465 
466     fi_moveCursorContinuous:
467 
468     mov rsp, rbp
469     pop rbp
470     ret
```

4.1.6 calcIndex

```
485 calcIndex:
486     push rbp
487     mov rbp, rsp
488 
489     ;Salvar registres a la pila
490     push rax
491     push rbx
492 
493     mov eax, DWORD [row]
494     imul eax, 7
495     mov bl, BYTE [col]
496     sub bl, 'A'
497     add eax, ebx
498     mov DWORD [pos], eax
499 
500     ;Recuperar registres de la pila
501     pop rbx
502     pop rax
503 
504     mov rsp, rbp
505     pop rbp
506     ret
```

4.1.7 putPiece

```
533     putPiece:
534     push rbp
535     mov rbp, rsp
536 
537     ;Salvar registres a la pila
538     push rax
```

```
539     push rbx
540
541     ;Inicialitzar registres
542     mov eax, 0
543     mov ebx, 0
544
545     call showBoard
546     call moveCursorContinuous
547
548     cmp BYTE [tecla], '␣'
549     jne fi_putPiece
550
551     mov al, BYTE [colCursor]
552     mov BYTE [col], al
553     mov DWORD [row], 5
554
555     bucle_putPiece:
556        call calcIndex
557        mov eax, DWORD [pos]
558        mov bl, BYTE [mBoard + eax] ;Accedim a l'índex de la matriu
559        cmp bl, '.' ;Si no hi ha res, podem col·locar la peça
560        je colocacion
561        cmp DWORD [row], 0;Si ja hi ha una fitxa, hem de mirar que no ens haguem passa
562        jl fi_putPiece
563        dec DWORD [row]
564        jmp bucle_putPiece
565
566     colocacion:
567        mov BYTE [mBoard + eax], 'X' ;Col·loquem la peça
568        call showBoard ;Mostrem l'estat actual de la taula
569
570     fi_putPiece:
571
572     ;Recuperar registres de la pila
573     pop rbx
574     pop rax
575
576     mov rsp, rbp
577     pop rbp
578     ret
```

4.2 Nivell mig

Nivel medio que guay

4.2.1 checkRow

```
524 checkRow:
525
526     push rbp
527     mov rbp, rsp
528
529     push rax
530     push rbx
531
532     mov rax, 0
533     mov rbx, 0
534     mov DWORD [inaRow], 0
535
536     mov eax, DWORD [pos] ;eax conté l'índex de l'última fitxa col·locada
537
538     mov bl, BYTE [mBoard + eax] ;eax conté l'offset de mBooard
539                           ;bl conté el char ('X' ó'O'). bl mai serà '.'
540
541     bucle_vertical:
542         cmp eax, 42;42 = últim índex de mBoard
543         jge fi_vertical
544         ;Si no ens hem sortit de la matriu...
545         cmp bl, BYTE [mBoard + eax]
546         jne fi_vertical
547         ;Si no ens hem sortit de la matriu
548         ;i els chars correlatius coincideixen...
549         add eax, 7;Anem a la següent fila
550         inc DWORD [inaRow] ;Incrementem el comptador
551         cmp DWORD [inaRow], 4
552         jne bucle_vertical
553         ;tas feli, has ganao
554         mov DWORD [row4Complete], 1;S'activa l'indicador
555
556     fi_vertical:
557
558     pop rbx
559     pop rax
560
561     mov rsp, rbp
562     pop rbp
563     ret
```

4.2.2 putPiece

```
770 putPiece:
771     push rbp
772     mov rbp, rsp
773
```

```
774     ;Salvar registres a la pila
775     push rax
776     push rbx
777
778     ;Inicialitzar registres
779     mov eax, 0
780     mov ebx, 0
781
782     call showBoard
783     call moveCursorContinuous
784
785     cmp BYTE [tecla], '␣'
786     jne fi_putPiece
787
788     mov al, BYTE [colCursor]
789     mov BYTE [col], al
790     mov DWORD [row], 5
791
792     bucle_putPiece:
793        call calcIndex
794        mov eax, DWORD [pos]
795        mov bl, BYTE [mBoard + eax] ;Accedim a l'índex de la matriu
796        cmp bl, '.' ;Si no hi ha res, podem col·locar la peça
797        je colocacion
798        cmp DWORD [row], 0;Si ja hi ha una fitxa, hem de mirar
799                          ;que no ens haguem passat dels límits de mBoard
800        jl fi_putPiece
801        dec DWORD [row]
802        jmp bucle_putPiece
803
804     colocacion:
805        cmp DWORD [player], 1
806        jne jugador2
807        mov BYTE [mBoard + eax], '0' ;Col·loquem la peça del jugador 1
808        jmp fi_putPiece
809        jugador2:
810           mov BYTE [mBoard + eax], 'X' ;Col·loquem la peça del jugador 2
811
812     fi_putPiece:
813        call showBoard ;Mostrem l'estat actual de la taula
814        call checkRow ;Comprovem si s'han encadenat 4fitxes iguals
815
816     ;Recuperar registres de la pila
817     pop rbx
818     pop rax
819
820     mov rsp, rbp
821     pop rbp
822     ret
```

4.2.3 put2Players

```
841 put2Players:
842     push rbp
843     mov rbp, rsp
844 
845     mov DWORD [player], 1;Forcem que primer jugui el jugador 1
846 
847     bucle_put2Players:
848         call showPlayer ;Mostrem el jugador
849         call putPiece ;Permetem la jugada
850         cmp BYTE [tecla], ' '
851         jne fi_put2Players ;Si no és l'espai, per força serà [ESC]
852 
853         ;Només hem de deixar jugar a player2 si player1 no ha guanyat
854         cmp DWORD [row4Complete], 1
855         je fi_put2Players
856         ;Evitem que player2 jugui infinitament
857         cmp DWORD [player], 2
858         je fi_put2Players
859         ;Si player1 no ha guanyat, passem el torn a player2
860         mov DWORD [player], 2
861         jmp bucle_put2Players
862 
863     fi_put2Players:
864 
865     mov rsp, rbp
866     pop rbp
867     ret
```

4.2.4 Play

```
880 Play:
881     push rbp
882     mov rbp, rsp
883 
884     bucle_Play:
885         call put2Players
886         cmp BYTE [tecla], ' '
887         jne fi_Play
888         cmp DWORD [row4Complete], 1
889         je fi_Play
890         jmp bucle_Play
891 
892     fi_Play:
893 
894     mov rsp, rbp
```

```
895     pop rbp
896     ret
```

4.3 Nivell avançat

Nivel avanzado que avanzado

4.3.1 checkRow

```
524 checkRow:
525 
526     push rbp
527     mov rbp, rsp
528 
529     ;Salvar registres a la pila
530     push rax
531     push rbx
532     push rcx
533     push rdx
534 
535     ;Inicialitzar registres
536     mov rax, 0
537     mov rcx, 0
538     mov rbx, 0
539     mov rdx, 0
540     mov DWORD [inaRow], 0
541     mov ecx, DWORD [pos] ;eax conté l'índex de l'última fitxa col·locada
542     mov bl, BYTE [mBoard + ecx] ;eax conté l'offset de mBooard
543                             ;bl conté el char ('X' ó'O'). bl mai
544                             ;serà '.' menys en un cas especific
545     cmp bl, '.'
546     je fi_vertical
547 
548     bucle_horizontal_adelante:
549         ;Si no ens hem sortit de la matriu...
550         cmp bl, BYTE [mBoard + ecx]
551         jne fi_bucle_horizontal_adelante
552         ;i els chars correlatius coincideixen...
553         inc ecx ;Anem a la següent columna
554         inc DWORD [inaRow] ;Incrementem el comptador
555         ;Això és per saber si ens passem de columna
556         cmp DWORD [inaRow], 4
557         je fi_row4 ;En cas que sigui 4, saltem a fi_row4
558         mov eax, ecx
559         push rcx
560         mov rdx, 0
561         mov ecx, 7
```

```
562         div ecx ;Aquesta operació divideix el que hi ha a eax
563                 ;(en aquest cas, la posició) per el nombre i
564                 ;emmagatzema a edx el reminder
565         pop rcx
566         cmp edx, 0
567         je fi_bucle_horizontal_adelante
568         ;En cas contrari, podem continuar
569         jmp bucle_horizontal_adelante
570
571     fi_bucle_horizontal_adelante:
572         mov ecx, DWORD [pos] ;eax conté l'índex de l'última fitxa
573                              ;col·locada
574         dec ecx ;Si comprovem de primeres el mateix, seríen 3i no 4
575
576     bucle_horizontal_atras:
577         cmp ecx, 0;0 = primer índex de mBoard
578         jl fi_bucle_horizontal_atras
579         ;Si no ens hem sortit de la matriu...
580         cmp bl, BYTE [mBoard + ecx]
581         jne fi_bucle_horizontal_atras
582         ;i els chars correlatius coincideixen...
583         dec ecx ;Anem a la següent columna
584         inc DWORD [inaRow] ;Incrementem el comptador
585         ;Això és per saber si ens passem de columna
586         cmp DWORD [inaRow], 4
587         je fi_row4 ;En cas que sigui 4, saltem a fi_row4
588         mov eax, ecx
589         push rcx
590         mov rdx, 0
591         mov ecx, 7
592         div ecx ;Aquesta operació divideix el que hi ha a eax
593                 ;(en aquest cas, la posició) per el nombre i
594                 ;emmagatzema a edx el reminder
595         pop rcx
596         cmp edx, 6
597         je fi_bucle_horizontal_atras
598         ;En cas contrari, podem continuar
599         jmp bucle_horizontal_atras
600
601     fi_bucle_horizontal_atras:
602         mov ecx, DWORD [pos] ;eax conté l'índex de l'última fitxa
603                              ;col·locada
604         mov DWORD [inaRow], 0
605
606     bucle_diagonal_adelante:
607         ;Si no ens hem sortit de la matriu...
608         cmp ecx, 42;42 = últim índex de mBoard
609         jge fi_bucle_diagonal_adelante
610         cmp bl, BYTE [mBoard + ecx]
```

```
611         jne fi_bucle_diagonal_adelante
612         ;i els chars correlatius coincideixen...
613         add ecx, 8;Anem a la següent columna
614         inc DWORD [inaRow] ;Incrementem el comptador
615         ;Això és per saber si ens passem de columna
616         cmp DWORD [inaRow], 4
617         je fi_row4 ;En cas que sigui 4, saltem a fi_row4
618         mov eax, ecx
619         push rcx
620         mov rdx, 0
621         mov ecx, 7
622         div ecx ;Aquesta operació divideix el que hi ha a eax
623                 ;(en aquest cas, la posició) per el nombre i
624                 ;emmagatzema a edx el reminder
625         pop rcx
626         cmp edx, 0
627         je fi_bucle_diagonal_adelante
628         ;En cas contrari, podem continuar
629         jmp bucle_diagonal_adelante
630
631     fi_bucle_diagonal_adelante:
632         mov ecx, DWORD [pos] ;eax conté l'índex de l'última fitxa
633                              ;col·locada
634         sub ecx, 8;Si comprovem de primeres el mateix, seríen 3i no 4
635
636     bucle_diagonal_atras:
637         cmp ecx, 0;0 = primer índex de mBoard
638         jl fi_bucle_diagonal_atras
639         ;Si no ens hem sortit de la matriu...
640         cmp bl, BYTE [mBoard + ecx]
641         jne fi_bucle_diagonal_atras
642         ;i els chars correlatius coincideixen...
643         sub ecx, 8;Anem a la següent columna
644         inc DWORD [inaRow] ;Incrementem el comptador
645         ; aixo es per saber si ens passem de columna
646         cmp DWORD [inaRow], 4
647         je fi_row4 ; en cas de que sigui 4, saltem a fi_row4
648         mov eax, ecx
649         push rcx
650         mov rdx, 0
651         mov ecx, 7
652         div ecx ;aquesta operació divideix el que hi ha a eax (en aquest cas, la posició)
653         pop rcx
654         cmp edx, 6
655         je fi_bucle_diagonal_atras
656         ; en cas contrari, podem continuar
657         jmp bucle_diagonal_atras
658
659     fi_bucle_diagonal_atras:
```

```
660         mov ecx, DWORD [pos] ;eax conté l'índex de l'última fitxa col·locada
661         mov DWORD [inaRow], 0
662
663     bucle_antidiagonal_adelante:
664         cmp ecx, 42;42 = últim índex de mBoard
665         jge fi_bucle_antidiagonal_adelante
666         ;Si no ens hem sortit de la matriu...
667         cmp bl, BYTE [mBoard + ecx]
668         jne fi_bucle_antidiagonal_adelante
669         ;i els chars correlatius coincideixen...
670         add ecx, 6;Anem a la següent columna
671         inc DWORD [inaRow] ;Incrementem el comptador
672         ; aixo es per saber si ens passem de columna
673         cmp DWORD [inaRow], 4
674         je fi_row4 ; en cas de que sigui 4, saltem a fi_row4
675         mov eax, ecx
676         push rcx
677         mov rdx, 0
678         mov ecx, 7
679         div ecx ;aquesta operació divideix el que hi ha a eax (en aquest cas, la posició)
680         pop rcx
681         cmp edx, 6
682         je fi_bucle_antidiagonal_adelante
683         ; en cas contrari, podem continuar
684         jmp bucle_antidiagonal_adelante
685
686     fi_bucle_antidiagonal_adelante:
687         mov ecx, DWORD [pos] ;eax conté l'índex de l'última fitxa col·locada
688         sub ecx, 6; Si comprovem de primeres el mateix, serien 3i no 4
689
690     bucle_antidiagonal_atras:
691         cmp ecx, 0;0 = primer índex de mBoard
692         jl fi_bucle_antidiagonal_atras
693         ;Si no ens hem sortit de la matriu...
694         cmp bl, BYTE [mBoard + ecx]
695         jne fi_bucle_antidiagonal_atras
696         ;i els chars correlatius coincideixen...
697         sub ecx, 6;Anem a la següent columna
698         inc DWORD [inaRow] ;Incrementem el comptador
699         ; aixo es per saber si ens passem de columna
700         cmp DWORD [inaRow], 4
701         je fi_row4 ; en cas de que sigui 4, saltem a fi_row4
702         mov eax, ecx
703         push rcx
704         mov rdx, 0
705         mov ecx, 7
706         div ecx ;aquesta operació divideix el que hi ha a eax (en aquest cas, la posició)
707         pop rcx
708         cmp edx, 0
```

```
709         je fi_bucle_antidiagonal_atras
710         ; en cas contrari, podem continuar
711         jmp bucle_antidiagonal_atras
712
713     fi_bucle_antidiagonal_atras:
714         mov ecx, DWORD [pos] ;eax conté l'índex de l'última fitxa col·locada
715         mov DWORD [inaRow], 0
716
717     bucle_vertical:
718         cmp ecx, 42;42 = últim índex de mBoard
719         jge fi_vertical
720         ;Si no ens hem sortit de la matriu...
721         cmp bl, BYTE [mBoard + ecx]
722         jne fi_vertical
723         ;i els chars correlatius coincideixen...
724         add ecx, 7;Anem a la següent fila
725         inc DWORD [inaRow] ;Incrementem el comptador
726         cmp DWORD [inaRow], 4
727         jne bucle_vertical
728
729     fi_row4:
730     mov DWORD [row4Complete], 1;S'activa l'indicador
731
732     fi_vertical:
733     ;Recuperar registres de la pila
734     pop rdx
735     pop rcx
736     pop rbx
737     pop rax
738
739     mov rsp, rbp
740     pop rbp
741     ret
```

5 Altres