

TITLE magicfar.asm: per esame 6/7/2005

COMMENT * Costruzione di un quadrato magico di ordine dispari $3 \leq n \leq 9$ (il valore di n scelto come esempio e' 7). Nota: con questo programma si possono generare quadrati magici di ordine anche superiore a 9: basta allocare lo spazio necessario cambiando il valore di N_max , e verificare che il numero $N_max * N_max - 1$ entri in una word. *

```
;-----  
;   C   O   S   T   A   N   T   I  
;-----
```

```
N_input equ 7           ; valore di n (dispari, in [3,9]) scelto come esempio  
N_max equ 9  
NxN_max equ N_max*N_max ; costante pre-calcolata dall'assemblatore
```

```
;-----  
;   S   T   A   C   K  
;-----
```

```
PILA SEGMENT STACK 'STACK'  
        DB      256 DUP('S')  
PILA ENDS
```

```
;-----  
;   D   A   T   A  
;-----
```

```
DATI SEGMENT PUBLIC 'DATA'  
MagicSquare      dw NxN_max dup(0)  
MSoffset          dw ?           ; sara' riempito dalla procedura con offset MagicSquare  
N                 dw ?           ; N_input  
C_N               dw ?           ; costante magica  
NM1               dw ?           ; N-1  
NM1_2             dw ?           ; 2*(N-1)  
N_2               dw ?           ; 2*N  
NxN               dw ?           ; N*N  
ix2N              dw N_max dup (?) ; vettore degli indici di riga pre-moltiplicati per N (vedi codice)  
DATI ENDS
```

```

;-----
;   C   O   D   E
;-----
CSEG SEGMENT PUBLIC 'CODE'

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;                               MAIN
;
MAIN PROC FAR
    ASSUME CS:CSEG,DS:DATI,SS:PILA,ES:NOTHING;

    MOV AX,SEG DATI
    MOV DS,AX

    ; salvataggio nello stack dei parametri da passare alla procedura
    mov ax,N_input
    push ax
    mov ax, offset MagicSquare
    push ax
    call far ptr CreateMagicSquare      ; N.B. la procedura non restituisce alcun valore
    add sp,4                          ; ripristina SP al valore che aveva prima della call

    mov ax,N_input*(N_input*N_input+1)/2 ; calcolo della costante magica (per il controllo)
    mov C_N,ax

check_magicsquare:                    ; il codice (omesso) che controlla il valore della
                                        ; costante per tutte le righe, le colonne e le diagonali
                                        ; va inserito qui

exit:
    MOV AH,4CH                        ; ritorno al DOS
    INT 21H

MAIN ENDP

```

```

,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
;
; CREATEMAGICSQUARE
;

```

```

CreateMagicSquare proc far

```

```

; N.B. Sia i l'indice di riga in word, e j quello di colonna (sempre in word). L'offset (in byte) della
; entry (i,j) nel buffer e'    bx=i*2*N+2*j=ix2N[si]+di    , dove si e di sono gli indici di riga e
; di colonna in byte. In sostanza, l'offset e' la somma di un offset di riga e di un offset
; di colonna. La singola entry e' indirizzabile come MagicSquare[bx].
; Il vettore ix2N[] conterra' gli offset di riga, pre-calcolati.

```

```

mov ax,[bp+4]      ; prelievo dallo stack di offset MagicSquare
mov MSoffset,ax
mov ax,[bp+6]      ; prelievo dallo stack di N_input
mov N,ax

```

```

; calcolo di 2*N, (N-1) e 2*(N-1)
mov N_2,ax
add N_2,ax
sub ax,1
mov NM1,ax
add ax,ax
mov NM1_2,ax

```

```

; Pre-calcolo degli offset di riga
xor ax,ax
xor bx,bx
xor si,si
mov cx,N

```

```

fill_ix2N:

```

```

mov ix2N[si],ax
add ax,N_2          ; Si tratta di indirizzi in un vettore di word
add bx,N
add si,2            ; Attenzione: ix2N e' un vettore di word, non di byte,
                    ; dunque l'indirizzo (in byte) dev'essere incrementato di 2

loop fill_ix2N
mov NxN,bx          ; all'uscita del ciclo bx contiene il valore N*N

```

```

; Riempimento del quadrato magico
;-----
mov si,0                ; L'elemento centrale della riga e' i=0,2*j=2*[(N-1)/2]=N-1
mov di,NM1
mov ax,1                ; AX conterra' via via i numeri da inserire
mov cx,NxN              ; Numero di iterazioni del ciclo
fill_MagicSquare:      ; Riempimento di Cold
    mov bx,MSoffset
    add bx,ix2N[si]
    add bx,di            ; BX contiene l'indice di riga nel buffer (vedi sopra)
    mov [bx],ax          ; Scostamento rispetto al segmento dati=offset MagicSquare + bx (vedi sopra)

    cmp si,0
    je check_upper_right_cell
    cmp di,NM1_2
    je last_column_no_first_row

no_last_column_no_first_row:
check_if_already_filled:
    sub si,2            ; Decremento ottimistico: anticipato qui, valido se Cnew vuota
    mov bx,MSoffset     ; Cnew e' gia' piena se non contiene il valore di inizializzazione 0
    add bx,ix2N[si]
    add di,2            ; Incremento ottimistico: anticipato qui, valido se Cnew vuota
    add bx,di
    mov dx,[bx]
    cmp dx,0
    jne already_filled
not_already_filled:
    jmp fill_new_cell_if_any

already_filled:
    add si,2            ; Compensa il decremento ottimistico ripristinando la riga di Cold
    add si,2            ; Avanza di una riga
    sub di,2            ; Ripristina la colonna di Cold
    jmp fill_new_cell_if_any

```

```

last_column_no_first_row:
    sub si,2
    mov di,0
    jmp fill_new_cell_if_any

check_upper_right_cell:
    cmp di,NM1_2
    jl first_row_no_last_column
upper_right_cell:
    add si,2 ; N.B.: in questo caso NON si incrementa di
    jmp fill_new_cell_if_any

first_row_no_last_column:
    mov si,NM1_2
    add di,2

fill_new_cell_if_any:
    inc ax ; Dopo l'incremento, AX contiene il nuovo numero da inserire
    dec cx
    cmp cx,0
    je return_to_caller
    jmp fill_MagicSquare

return_to_caller:
    ret ; N.B. lo stack pointer va ripristinato al valore precedente la call
    ; con 2 pop oppure incrementandolo di 4

CreateMagicSquare endp

CSEG ENDS

END MAIN ; il programma comincia all'indirizzo di MAIN

```