FUNCIONES DE GESTIÓN DE ARCHIVOS	OBSERVACIONES
(prin1 expr descriptor_archivo) **(princ expr descriptor_archivo)** **(print expr descriptor_archivo)**	• Ver consideraciones en capítulo 11, Funciones de control de pantalla.
(read-line descriptor_archivo) *Lee una cadena del teclado o de un archivo abierto*	• Siendo f un descrptor de archivo abierto válido, (read-line f) devuelve la siguiente línea de entrada del archivo con formato de cadena, o nil si ha llegado al final del archivo.
(write-char núm descriptor_archivo)	• El argumento núm es el código ASCII decimal del carácter que debe escribirse, así como el valor devuelto por write-char.
(write-line cadena descriptor_archivo) *Escribe una cadena en la pantalla o en un archivo abierto*	• Devuelve la cadena entrecomillada de la forma habitual, pero omite las comillas cuando la escribe en un archivo.
(close desc_arch) *Cierra un archivo abierto*	• El argumento descriptor_archivo es un descriptor de archivo que se obtiene mediante la función open. Tras utilizar la función close, aunque el descriptor del archivo no cambia, deja de ser válido. Los datos añadidos a un archivo abierto no se escriben realmente hasta que se cierra el archivo. La función close devuelve nil si descriptor_archivo es válido, en caso contrario devuelve un mensaje de error.
(findfile nombre_archivo) *Busca el camino de la biblioteca AutoCAD para el archivo especificado*	• La función findfile no asigna el tipo o extensión del archivo especificado en nombre_archivo. • Si nombre_archivo no especifica un prefijo de unidad/directorio en este argumento, findfile busca en el camino de la biblioteca de AutoCAD. • Si se escribe un prefijo de unidad/directorio, findfile busca en dicho directorio. • La función findfile siempre devuelve un nombre completo de unidad/directorio/archivo o nil si el archivo especificado no se ha encontrado.
(open nombre_archivo modo) *Abre un archivo para que accedan a él las funciones de E/S de AutoLISP*	• El argumento nombre_archivo es una cadena que especifica el nombre y la extensión del archivo que debe abrirse. El argumento modo es la etiqueta de lectura/escritura y debe contener una cadena de una sola letra en minúsculas: **"r"** para leer, **"a"** para escribir añadiendo o **"w"** para escribir desde el principio. • Si en modo "w" o "a" el archivo no existe, se crea. • Los datos que se pasan a un archivo abierto no se escriben realmente hasta que se cierra e l archivo con la función close. • Open devuelve un descriptor de archivos para que lo usen otras funciones de E/S. El descriptor debe asignarse a un símbolo que utilice la función setq.
(read-char descriptor_archivo) *Devuelve el código ASCII decimal que representa el carácter leído en el búfer de entrada por teclado o en un archivo abierto*	• Si no se especifica descriptor_archivo y el búfer de entradas por teclado carece de caracteres, read-char espera a que se escriban datos desde el teclado (seguidos de RETURN). Por ejemplo, si se tiene en cuenta que el búfer de entradas por teclado está vacío, (read-char) espera a que se escriban datos. Si introduce los caracteres ABC seguidos de RETURN, read-char devuelve 65 (el código ASCII decimal correspondiente a la letra A). Las tres llamadas siguientes a read-char devuelven 66, 67 y 10 (línea nueva), respectivamente. Si se efectúa otra llamada, read-char vuelve a esperar a que se escriban datos.

EJEMPLOS DE FUNCIONES DE GESTIÓN DE ARCHIVOS

; Considérese de nuevo la variable l valorada con la siguiente lista de puntos:
l = ((104.489 186.88 0.0) (238.529 185.446 0.0) (149.89 178.272 0.0) (179.436 177.554 0.0) (163.582 108.685 0.0) (104.489 161.772 0.0))

```
( setq f1 ( open "c:/cicyt/capítulo 12/fich1.dat" "w" ) )
        ( setq i 0 sw nil)
        ( while ( < i ( length puntos ) )
                ( if sw
                        ( print ( nth i puntos ) f1 )
                        ( progn
                                ( prin1 ( nth i puntos ) f1 )
                                ( setq sw t )
                        ); fin del progn
                ); fin del if
                ( setq i ( + 1 i ) )
        ); fin del while

        ( close f1 )
```

FICH1.DAT
(104.489 186.88 0.0)
(238.529 185.446 0.0)
(149.89 178.272 0.0)
(179.436 177.554 0.0)
(163.582 108.685 0.0)
(104.489 161.772 0.0)
MARCA DE FIN DE FICHERO

El fragmento propuesto crea el fichero de nombre FIH1.DAT en el directorio indicado y lo rellena con los datos que se ofrecen en la tabla adjunta, a razón de un punto por cada registro:

- Inicialmente, se abre el fichero con la función *open*.

Se establece un ciclo mientras el contador i sea menor que el total de puntos de la lista l (la comparación es un menor estricto porque la función *nth* comienza a obtener puntos de la lista desde cero).

- La función *If* tiene una rama de salida THEN que escribe usando *print* y una rama ELSE que escribe con *prin1* y valora la variable SW = NIL.
- Con eso se consigue que en la primera vuelta se escriba con *prin1* y en todas las demás con *print*. Dado que la función *print* salta el registro antes de escribir, este recurso permite no dejar un registro en blanco al escribir el fichero.

En la parte inferior de la página se muestra el código preciso para efectuar la operación inversa: leer el fichero e integrar todos sus componentes en una lista de nombre “puntos” con una estructura y contenido idénticos a los de la lista “l” del código anterior. Una vez abierto el fichero en modo lectura, la función *read-line* lee registros completos y los devuelve a la corriente del programa en forma de alfanuméricos; la función *read* elimina las comillas, convirtiendo los valores automáticamente en estructuras de lista, tras lo cual la orden *cons* va generando la lista puntos definitiva.
La lectura de la marca fin de fichero devuelve NIL, lo que provoca la salida del ciclo con la lista puntos ya generada.

```
( setq f1 ( open "c:/cicyt/capítulo 12/fich1.dat" "r" ) )
( setq punto nil puntos nil )

( while ( setq punto ( read-line f1 ) )
        ( setq puntos ( cons ( read punto ) puntos ) )
)
```

TÉCNICA BÁSICA 1 PARA ESCRIBIR ARCHIVOS.

LISTA → ARCHIVO

<Sin título-0>

```
(defun captura_puntos (/ punto)
  (setq puntos nil)
  (while (setq punto (getpoint))
    (setq puntos (cons punto puntos))
  )
  (length puntos)
)
; *************************************************************
(defun c:ej3 ()
  (setq npuntos (captura_puntos))
  (setq f1 (open "c:/cicyt/capítulo 12/fich1.dat" "w"))
  (setq i  0
        sw nil
  )
  (while (< i npuntos)
    (if sw
      (print (nth i puntos) f1)
      (progn
        (prin1 (nth i puntos) f1)
        (setq sw t)
      ) ; fin del progn
    )   ; fin del if
    (setq i (+ 1 i))
  )     ; fin del while
  (close f1)
)
```

CONSIDERACIONES FUNDAMENTALES:

1) Apertura del archivo. OPEN en modo “w” (MINÚSCULA)
2) Direccionamiento: la barra separadora es la convencional “/” , no “\”
3) Descriptor del fichero: concepto
4) Diferenciar claramente el cometido de PRIN1 y de PRINT
5) Diferenciar también el cometido de PRIN1 y de PRINC
6) Asociar WRITE-LINE con PRINC
7) Cerrado de fichero: CLOSE

TÉCNICA BÁSICA 2, PARA LEER ARCHIVOS.

ARCHIVO → LISTA

<Sin título-0>

```
( defun dibuja_curva ( puntos )
( command "pol" )
( while ( setq p ( car puntos ) )
( command p )
( setq puntos ( cdr puntos ) )
)
( command "" )
)
; ************************************************************
( defun c:ej4 ()
( setq f1 ( open "c:/cicyt/capítulo 12/fich1.dat" "r" ) )
( while ( setq punto ( read-line f1 ) )
( setq puntos ( cons ( read punto ) puntos ) )
)
( dibuja_curva puntos )
)
```

CONSIDERACIONES FUNDAMENTALES:

1) Apertura del archivo. OPEN en modo “r” (MINÚSCULA)
2) Funcionalidad de READ-LINE
3) Papel de READ en el ciclo
4) Archivos cuyos registros no son listas: STRCAT “(“ …… “)”
5) No olvidar CERRAR los ficheros