7.1.- LECTURA Y ESCRITURA POR FICHERO

En el capitulo 2 , en el primer programa escrito que se presentó llamado AREAS, se conocieron por primera vez las siguientes sentencias FORTRAN:

```
      PRINT*,´DAME EL LADO a´
      READ*,a
```

La sentencia **PRINT*,** nos ha permitido presentar por pantalla mensajes, datos y resultados durante la ejecución del programa de manera muy simple; por otro lado la sentencia **READ*,** facilitaba la lectura por pantalla de datos de cualquier tipo, reales, enteros o de carácter. Ambas sentencias se han utilizado sin conocer en profundidad su estructura, sin embargo nos han servido para construir programas e ir aprendiendo otras sentencias FORTRAN no relacionadas con la lectura y escritura de información. Ahora desarrollaremos de manera mas profunda este tipo de sentencias FORTRAN las cuales toman su verdadera importancia cuando en vez de realizar por pantalla la lectura y escritura de datos se hace a través de ficheros.

Si recordamos el ejemplo del capítulo anterior en el que los resultados de los números de teléfono se presentaban por pantalla agrupados por pisos, es natural preguntarse como seriamos capaces de visualizar los resultados y comprobar su validez para el caso de un edifico de 30 plantas, esta claro que nos seria imposible ver por pantalla todos los resultados. Además aunque fuéramos capaces de verlos, los resultados no habrían quedado registrados en ningún lugar y para volver a verlos deberíamos realizar una nueva ejecución del programa. Este procedimiento esta claro que únicamente es valido para cálculos sencillos propios de una calculadora de bolsillo pero cuando los datos de entrada y los de salida son numerosos es preciso recurrir a métodos de almacenamiento seguros y rápidos.

FORTRAN para resolver este problema permite incluir una serie de sentencia en el programa que son capaces de:

1. ***ABRIR UN FICHERO***
2. ***LEER SU CONTENIDO O ESCRIBIR NUEVA INFORMACION***
3. ***CERRAR EL FICHERO***

Ahora vamos a ir describiendo cada una de las 3 acciones anteriormente descritas:

1. ABRIR UN FICHERO *SENTENCIA* : OPEN

Recordemos el programa TELFO del apartado 6.2 del capitulo 6, en él los números de teléfono de los vecinos eran asignados dentro del propio programa, a continuación se muestran las sentencias empleadas:

```
C     CARGO EL ARRAY
      TF(1,1)=2157
      TF(1,2)=2557
```

Esta opción tiene un inconveniente y es que si uno de los vecinos cambia de numero de teléfono será preciso cambiarlo en el fichero de código fuente, compilarlo y construir el ejecutable nuevamente. Otra opción que podríamos haber planteado es haber leído los números de teléfono por pantalla cada vez que ejecutáramos el programa; sin embargo, esto hubiera complicado enormemente cada ejecución. La solución ideal esta en crear un fichero de datos independiente al que podemos llamar DATOS.DAT y en el escribimos los números de teléfono tal y como se describe a continuación:

FICHERO: **DATOS.DAT**

```
2,1,5,6,7,7
2,5,7,5,5,7
2,1,5,4,7,7
2,5,5,1,8,7
```

2	8	5	3	4	7																							
2	9	4	6	3	7																							
2	8	5	3	4	7																							
2	3	5	7	3	7																							

Este fichero escrito en código ASCII le podremos crear con cualquier editor de texto que tengamos a nuestra alcance y puede hacerse cómodamente desde el propio editor del entorno de programación FORTRAN, de la misma forma que si estuviéramos creando un nuevo programa.

Como puede verse el fichero únicamente contiene los números de teléfono de los vecinos, uno por línea, comenzando en la posición 1 de la cada fila.

EJERCICIO 7.1

- Crear el fichero DATOS.DAT.

Una vez creado el fichero con los números de teléfono y guardado con el nombre DATOS.DAT podemos pensar en utilizarle mas adelante como fuente de información y podemos ir modificando el programa original TELEF hasta llegar a uno nuevo que lea los teléfonos por fichero y presente los resultados en otro fichero. Para ello deberemos en primer lugar incluir el programa la sentencia OPEN para ABRIR dicho fichero.

La sentencia OPEN tiene la siguiente estructura:

SENTENCIA	DESCRIPCION DE OPCIONES MAS IMPORTANTES
OPEN (opciones)	**1. UNIT = ee** En donde **ee** es una expresión entera que indica la unidad de entrada y salida a la que debe conectarse el fichero. Este parámetro es obligatorio y siempre debe aparecer a continuación del paréntesis, es decir el primero. **2. FILE = nom** En donde **nom** es una expresión de tipo carácter que identifica el nombre del fichero que se desea abrir. (NOTA: Por regla general el fichero que se abre desde un programa estará presente en el directorio desde el que se esta ejecutando el propio programa, de no ser así **nom** deberá indicar además del nombre del fichero el camino de acceso al fichero.) **3. STATUS = tp** En donde **tp** es una expresión tipo carácter que puede tomar los siguientes valores: '**NEW**': Significa que el fichero que se va a abrir no ha sido creado anteriormente y por lo tanto la sentencia OPEN lo creará, esta opción esta pensada para ficheros de resultados y no de datos. (NOTA: si el fichero existiera el programa provocara un error de ejecución) '**OLD':** Al contrario que NEW, significa que el fichero deberá estar creado con anterioridad a su apertura, esta opción es propia de ficheros de datos. (NOTA: si el fichero no existe, el programa provocara un error) '**UNKNOW**' : Esta opción es un compendio de las dos anteriores y suele ser muy útil en etapa de pruebas ya que es el programa el que se encarga de buscar el fichero y en caso de estar presente le asigna el valor OLD y en caso de no estar presente le considerara como NEW y le creará.

	'**SCRATCH**' : Caso especial de STATUS para poder definir un fichero temporal de trabajo que podrá desaparecer al finalizar el programa.

Ejemplos:

SENTENCIA	ACCION
OPEN(UNIT=10, FILE='DATOS.DAT')	Abre un fichero con dirección de unidad 10 y de nombre DATOS.DAT
OPEN(20,FILE='RESUL',STATUS='NEW')	Abre un fichero con dirección de unidad 20 y de nombre RESUL, además el fichero será considerado como nuevo y por lo tanto se creará.
OPEN(M+2, FILE='CCC',STATUS='UNKNOW')	Abre un fichero con dirección de unidad M+2 y de nombre CCC, además el estado del fichero será considerado como desconocido y por lo tanto se creará en caso de no estar presente.

Una vez vista la construcción de la sentencia OPEN, podemos ya establecer que para el ejemplo de los números de teléfono la sentencia adecuada para abrir el fichero DATOS:DAT será:

OPEN (10, FILE='DATOS.DAT',STATUS='OLD')

Por otro lado hemos comentado anteriormente que también desearíamos obtener los resultados por fichero el cual puede ser creado automáticamente por el programa cada vez que se ejecute y así poder visualizar los resultados en cualquier momento después de ser ejecutado el programa. Para lograr esto la sentencia que será preciso incluir en el programa será:

OPEN (20, FILE='RESUL.RES',STATUS='NEW')

Con lo anterior conseguimos dos propósitos abrir dos ficheros uno de datos y otro de resultados, ahora habrá que leer en el primero y escribir en el segundo.

2.1 ESCRIBIR EN UN FICHERO *SENTENCIA* : **READ**

Hasta el momento la sentencia que se ha empleado para leer datos por pantalla ha sido la sentencia READ*, esta es una simplificación de la sentencia READ completa que ahora se va a presentar. La nueva forma de usar READ permite además de la lectura de datos por pantalla la lectura de datos por fichero. Su construcción y descripción se muestran a continuación:

SENTENCIA	DESCRIPCION DE OPCIONES MAS IMPORTANTES
READ (opciones) Lista	**1. UNIT = ee** En donde **ee** es una expresión entera que indica la unidad de entrada y salida del fichero del que se extraerán los datos. Este parámetro es obligatorio y siempre debe aparecer a continuación del paréntesis, es decir el primero. Con este parámetro relacionaremos la sentencia OPEN y READ para que la lectura se produzca en el fichero deseado. Si este parámetro es sustituido por un *, significara que la lectura será por pantalla y por lo tanto equivaldrá a READ*, la cual ya la conocemos.

	2. FMT = f En donde **f** puede ser un entero que indica el número de etiqueta en el que se encuentra especificado el formato de lectura establecido. Si este parámetro es sustituido por un *, indicara que la lectura se va hacer con formato libre, es decir igual que como se venia haciendo cuando era utilizada la sentencia READ*. Aun no se han mostrado los diferentes formatos de lectura y escritura que admite FORTRAN, sin embargo se adelanta que este tipo de sentencias permiten establecer el tipo de dato (real, entero, carácter, etc.) y su extensión (longitud) que se desea leer o escribir en un fichero o en pantalla. En los ejercicios resueltos con anterioridad se habrá podido observar que el programador no tenia excesivo control en cuanto a la forma de presentación de los resultados. Esto cambiará radicalmente mediante con la utilización de los formatos que FORTRAN permite. La fijación definitiva de los formatos de lectura y escritura suele definirse en la etapa final de programación en la que las tareas de depuración del código y su puesta apunto son realizadas. **Lista :** Aquí deberán aparecer los nombres del conjunto de variables que se van a leer que como ya es sabido deberán estar separadas por comas.

Ejemplos:

SENTENCIA	ACCION
READ(UNIT=10, *)A,B	Se sitúa en el fichero con dirección de unidad 10 y lee A y B con formato libre. Para conocer el nombre del fichero habrá que buscar la sentencia OPEN con UNIT=10
READ(20,200)DAT1	Se sitúa en el fichero con dirección de unidad 20 y leerá la variable DAT1 con el formato especificado en la sentencia de formato con etiqueta 200. *(Esto último se verá en el próximo capítulo).*
READ(*,*)AS	Leerá la variable AS por pantalla y con formato libre. Esta sentencia equivale a: **READ*,AS**

Una vez conocida la sentencia READ podremos programar la lectura del fichero DATOS.DAT creado previamente, para ello bastaría incluir las siguientes líneas de código:

Recordando que ya hemos incluido la sentencia: `OPEN (10,FILE='DATOS.DAT',STATUS='OLD')`

La lectura sería:

```
DO I=1,4
      DO J=1,2
            READ(10,*)TF(I,J)
      ENDDO
ENDDO
```

2.2 ESCRIBIR EN UN FICHERO *SENTENCIA* : **WRITE**

Hasta el momento la sentencia empleada para presentar datos por pantalla ha sido sentencia PRINT, esta es una simplificación de la sentencia WRITE que ahora se va a presentar. WRITE presenta todas las capacidades que nos ofrece PRINT pero además permite la escritura de datos en fichero. Su construcción y descripción presenta las mismas opciones que fueron descritas para la sentencia READ, a continuación se muestran:

SENTENCIA	DESCRIPCION DE OPCIONES MAS IMPORTANTES
WRITE (opciones) Lista	**1. UNIT = ee** En donde **ee** es una expresión entera que indica la unidad de entrada y salida del fichero en el que se escribirán los datos. Este parámetro es obligatorio y siempre debe aparecer a continuación del paréntesis, es decir el primero. Con este parámetro relacionaremos la sentencia OPEN y WRITE para que la escritura se produzca en el fichero deseado. Si este parámetro es sustituido por un *, significara que la escritura será por pantalla y por lo tanto equivaldrá a PRINT*, la cual ya la conocemos. **2. FMT = f** En donde **f** puede ser un entero que indica el numero de etiqueta en el que se encuentra especificado el formato de escritura establecido. Si este parámetro es sustituido por un *, indicara que la escritura se va hacer con formato libre, es decir igual que como se venia haciendo cuando era utilizada la sentencia PRINT*. Aun no se han mostrado los diferentes formatos de lectura y escritura que admite FORTRAN, sin embargo se adelanta que este tipo de sentencias permiten establecer el tipo de dato (real, entero, carácter, etc.) y su extensión (longitud) que se desea leer o escribir en un fichero o en pantalla. En los ejercicios resueltos con anterioridad se habrá podido observar que el programador no tenia excesivo control en cuanto a la forma de presentación de los resultados. Esto cambiara radicalmente mediante con la utilización de los formatos que FORTRAN permite. La fijación definitiva de los formatos de lectura y escritura suele definirse en la etapa final de programación en la que la depuración del código y su puesta apunto son realizadas. **Lista :** Aquí deberán aparecer los nombres del conjunto de variables que se van a escribir que como ya es sabido deberán estar separadas por comas.

Ejemplos:

SENTENCIA	**ACCION**
WRITE(UNIT=10, *)A,B	Se sitúa en el fichero con dirección de unidad 10 y escribe A y B con formato libre. Para conocer el nombre del fichero habrá que buscar la sentencia OPEN con UNIT=10
WRITE(20,200) (A(I), I=1,5)	Se sitúa en el fichero con dirección de unidad 20 y escribirá los datos almacenados en la variable A(I) del 1 al 5, con el formato especificado en la sentencia de formato con etiqueta 200. *(Esto último se verá en el próximo capítulo).*
WRITE(*,*)AS	Escribirá la variable AS por pantalla y con formato libre. Esta sentencia equivale a: **PRINT*,AS**

Una vez conocida la sentencia WRITE podremos programar la escritura de datos sobre el fichero RESUL.DAT, para ello bastaría incluir las siguientes líneas de código:

Recordando que ya hemos incluido la sentencia: `OPEN (20,FILE='RESUL.DAT',STATUS='NEW')`

La escritura seria:

```
DO I=1,4
      WRITE(20,*)'PISO =',I
      DO J=1,2
            WRITE(20,*)TF(I,J)
      ENDDO
ENDDO
```

3 CERRAR UN FICHERO *SENTENCIA* : **CLOSE**

Hasta el momento hemos abierto ficheros, leído datos y escrito resultados, cuando estas dos últimas tareas han finalizado es recomendable cerrar los ficheros para ello se emplea la sentencia CLOSE cuya descripción se muestra a continuaron:

SENTENCIA	DESCRIPCION DE OPCIONES MAS IMPORTANTES
CLOSE (opciones)	**1. UNIT = ee** En donde **ee** es una expresión entera que indica la unidad de entrada y salida del fichero en el que se escribirán los datos. Este parámetro es obligatorio y siempre debe aparecer a continuación del paréntesis, es decir el primero. **2. STATUS = st** En donde **st** es una expresión carácter con dos posibilidades: 'KEEP' Esta opción indica que el fichero deberá guardarse (no destruirse) una vez cerrado, es una opción incompatible con los ficheros de tipo SCRATCH o temporales. 'DELETE' Indica que el fichero será borrado al finalizar el programa su ejecución. En caso de nos ser incluida la opción STATUS, la opción KEEP será la utilizada por FORTRAN. Por otro lado los ficheros de tipo SCRATCH son borrados automáticamente al finalizar el programa.

Ejemplos:

SENTENCIA	**ACCION**
CLOSE(UNIT=10)	Se cierra el fichero con número de unidad 10. Para conocer el nombre del fichero habrá que buscar la sentencia OPEN con UNIT=10
CLOSE(20,STATUS='DELETE')	Se cierra el fichero con número de unidad 20 y se borra.

Por ultimo, es preciso conocer algunos comportamientos de interés del conjunto de sentencias OPEN y CLOSE :

- Un fichero puede abrirse y cerrarse varia veces a lo largo de un programa.

- Cuando un fichero se habre para leer su contenido la posición inicial de lectura siempre se corresponde con la fila 1 y la columna 1. En caso de cerrarse y volverse a abrir, la posición de lectura vuelve a la posición inicial.

Siguiendo con el ejemplo en curso, para cerrar los ficheros DATOS.DAT y RESULT.RES bastaría incluir las siguientes líneas de código:

```
CLOSE(10)
CLOSE (20)
```

Con todo lo anterior el programa TELFO quedaría de la siguiente manera:

```
      PROGRAM  TELFO
      INTEGER  I,J,TF(4,2)
C   ABRIMOS FICHEROS
      OPEN(10,FILE='DATOS.DAT',STATUS='OLD')
      OPEN(20,FILE='RESUL.RES',STATUS='NEW')
C   LEEMOS NUMEROS DE TELEFONO
      DO I=1,4
            DO J=1,2
            READ(10,*) TF(I,J)
            ENDDO
      ENDDO
C   ESCRIBIMOS NUMEROS DE TELEFONO
      DO I=1,4
            WRITE(20,*)'PISO=',I
            DO J=1,2
            WRITE(20,*)'TELEFONO=',TF(I,J)
            ENDDO
      ENDDO
C   CERRAMOS FICHEROS
      CLOSE(10)
      CLOSE(20)
      END
```

Se recomienda copiar el programa anterior, ejecutarlo y comprobar los resultados abriendo el fichero de resultados RESUL.RES que habrá sido creado automáticamente tras la ejecución del código.

EJERCICIO 7.2

- Modificar el programa correspondiente al ejercicio 5.3 para que realice todas sus entradas y salidas a través de ficheros (uno de datos y otro de resultados).

EJERCICIO 7.3

- Crear un programa que lea un fichero de nombre NUM1.DAT que contenga N números reales, que los ordene de menor a mayor y escriba la vieja y la nueva ordenación en un nuevo fichero llamado RES1.RES.

  NOTA IMPORTANTE: Para que el programa conozca **N** es decir el número de números que contiene el fichero de datos, deberá leer en primer lugar **N(entero)** advirtiéndole del número de números que tiene que leer a continuación.

EJERCICIO 7.4

- Crear un programa que genere un fichero en el que se presente la tabla de multiplicación del 1 al 10, de manera comprensible.

EJERCICIO 7.5

- Crear un programa que lea un fichero de nombre NUM2.DAT que contenga 10 números reales, y que indique por pantalla cuantos numero pares hay, cuantos impares.

EJERCICIO 7.6

- Supongamos que tenemos un fichero de datos A con cuatro números enteros arbitrarios a modo de cuatro números secretos. Por otro lado tenemos otro fichero de datos B en el que introducimos 10 combinaciones de cuatro números enteros intentando acertar con la combinación secreta. Crear un programa que nos escriba en un fichero de resultados los aciertos encontrados en posición y coincidencia de valor entre cada conjunto de cuatro números y la combinación secreta.

EJERCICIO 7.7 (opcional)

- Mejorar el programa 7.6 para que se convierta en un juego interactivo, de tal forma que la combinación secreta estará en un fichero de datos, el usuario ira introduciendo por pantalla combinaciones de 4 números enteros y el programa ira mostrando por pantalla los aciertos. Además se irán escribiendo cada uno de los resultados parciales en un fichero de resultados.

7.2.- SENTENCIAS END FILE, REWIND Y BACKSPACE

Este tipo de sentencia permiten realizar determinadas acciones sobre los ficheros relacionadas con las entradas y salidas de datos.

- **END FILE**

SENTENCIA	DESCRIPCION DE OPCIONES MAS IMPORTANTES
END FILE (opciones)	**1. UNIT = ee** En donde **ee** es una expresión entera que indica la unidad de entrada y salida del fichero en el que se escribirán los datos. Este parámetro es obligatorio y siempre debe aparecer a continuación del paréntesis, es decir el primero.
ACCION REALIZADA	
Esta sentencia escribe en el fichero de unidad **ee** una marca de fin de fichero, indicando el final de los datos. ***Ejemplo:*** END FILE (20)	

- **REWIND**

SENTENCIA	DESCRIPCION DE OPCIONES MAS IMPORTANTES
REWIND(opciones)	**1. UNIT = ee** En donde **ee** es una expresión entera que indica la unidad de entrada y salida del fichero en el que se escribirán los datos. Este parámetro es obligatorio y siempre debe aparecer a continuación del paréntesis, es decir el primero.
ACCION REALIZADA	
Esta sentencia permite que después de leer los datos de un fichero y antes de proceder a su cierre mediante la sentencia CLOSE, el dispositivo de lectura o escritura se posiciona nuevamente en la fila 1 columna 1, es decir la posición de inicio. Esto permitiría volver a leer el fichero completamente de principio a fin, por si fuera necesario. Es importante resaltar que esta orden equivaldría a cerrar el fichero y abrirlo de nuevo, ya que al abrir un fichero el dispositivo se posiciona siempre en la posición inicial. Sin embargo esta ultima opción no es la mas recomendable si se desea leer varias veces un mismo fichero en una ejecución. ***Ejemplo:*** REWIND (20)	

- **BACKSPACE**

SENTENCIA	DESCRIPCION DE OPCIONES MAS IMPORTANTES
BACKSPACE(opciones)	**1. UNIT = ee** En donde **ee** es una expresión entera que indica la unidad de entrada y salida del fichero en el que se escribirán los datos. Este parámetro es obligatorio y siempre debe aparecer a continuación del paréntesis, es decir el primero.

ACCION REALIZADA
La acción realizada por BACKSPACE es equivalente a REWIND con la diferencia de que BACKSPACE únicamente retrocede un registro, es decir supongamos que acabamos de leer un primer dato en la primera línea y un segundo dato en la segunda línea de un fichero, si a continuación ejecutamos la sentencia BACKSPACE la próxima lectura se producirá nuevamente sobre el segundo dato de la segunda línea del fichero. La acción se comportaría de manera equivalente si estuviéramos escribiendo en vez de leyendo. ***Ejemplo:*** REWIND (20)

7.3.- OPCIONES ESPECIALES

Como hemos visto las sentencias descritas en el presente capitulo permiten una serie de opciones las cuales van encerradas entre paréntesis y establecen la forma de actuar de la propia sentencia. Las que han sido descritas en cada apartado son las mas importantes y las que con mayor seguridad van ser utilizadas de manera continuada, sin embargo existen otras opciones poco usuales pero que permiten determinadas acciones que en determinados momentos pueden ser de interés. A continuación se muestran dichas opciones, su descripción y para que sentencias son validas:

- **ERR**

OPCION	**DESCRIPCION**
ERR= ETIQUETA DE SENTENCIA	Permite que en caso de producirse un error en la acción que la sentencia que la contiene, el programa no se interrumpa y salte hasta la línea del programa que contiene la etiqueta de continuación. *SENTENCIAS QUE ADMITEN ESTA OPCION:* **READ, WRITE, OPEN, CLOSE, END FILE, REWIND y BACKSPACE**

Ejemplo:

```
      OPEN (20, FILE='XXXX', STATUS=' OLD', ERR=120)
       ..............................................
       ....líneas de programa.......................
        .............................................

120   CONTINUE
```

En el ejemplo anterior, si el programa intenta abrir el fichero XXXX y no lo encuentra, se producirá un error ya que al tener STATUS=OLD el programa considera que el fichero debería estar creado con anterioridad a la ejecución. Sin embargo el programa no se parara e ira a la línea en la que se encuentra la sentencia de continuación CONTINUE con etiqueta 120. Idéntica función se realizaría para las de mas sentencias que admiten esta opción.

- **END**

OPCION	DESCRIPCION
END= ETIQUETA DE SENTENCIA	Permite una bifurcación a una sentencia con etiqueta cuando se ha finalizado la operación de entrada o salida de datos. *SENTENCIAS QUE ADMITEN ESTA OPCION:* **READ, WRITE y END FILE**

Ejemplo:

```
      READ (10,END=120)A,B,C
       ..........................................................
       ....líneas de programa.....................................
       ..........................................................
120    CONTINUE
```

En el ejemplo anterior, el programa una vez leídos los datos A,B,C se bifurcara a la línea de programa que tiene la etiqueta de sentencia 120.

- **IOSTAT**

OPCION	DESCRIPCION
IOSTAT= NOMBRE DE VARIABLE ENTERA	Esta opción permite detectar el final de un fichero o la aparición de un error durante su procesado. IOSTAT podrá tomar tres valores diferentes dependiendo de: 1.- Valor 1 si se detecta un error. 2.- Valor 2 si se alcanza el final del fichero. 3.- Valor 3 Si no se produce ninguna de las dos situaciones anteriores. Los valore 1,2 y 3 dependen de la instalación y por ello deberán comprobarse con anterioridad, o mediante una ejecución dedicada a tal efecto.. *SENTENCIAS QUE ADMITEN ESTA OPCION:* **READ, WRITE, OPEN, CLOSE, END FILE, REWIND y BACKSPACE**

Ejemplo:

```
      READ (UNIT=10,FMT=210, IOSTAT=EST)X,Y,Z,M
```

En el ejemplo anterior, el programa leerá X,Y,Z y M en la unidad 10 con el formato especificado en la sentencia de etiqueta 210. En caso de haber algún tipo de error EST tomara el 1º valor, en caos de leerse todo hasta el final EST tomara el 2º valor y en caso de no producirse ninguna de las situaciones anteriores EST tomará el 3º valor.