8.1.- ENTRADA Y SALIDA CON FORMATO

El presente capitulo va a estar dedicado exclusivamente a presentar las diferentes maneras que FORTRAN ofrece para leer y escribir datos de cualquier tipo mediante un formato establecido por el programador. Las sentencias READ*, y PRINT*, ya han sido aplicadas reiteradamente en capítulos precedentes y su funcionamiento es conocido. Para ambas sentencias la lectura y escritura se realizan de una manera libre (sin formato) y por lo tanto el usuario únicamente tiene presente el tipo (real, entero, etc.) del dato que el ordenador espera procesar; sin embargo, hasta el momento nunca se ha fijado, por ejemplo, cuantos decimales debían aparecer en el resultado o si el dato leído estaba expresado en forma exponencial. Todos estos aspectos van a ser ahora presentados y una vez finalizado el capítulo se tendrá un control total sobre la forma de la información de entrada y de salida en un programa FORTRAN.

• LECTURA Y ESCRITURA SIN FORMATO

Recordando las sentencias READ* y PRINT*, sin formato, para la lectura y escritura de un numero real por pantalla tenemos que:

SENTENCIA	ACCION
READ *,A	• Leerá una variable **A** con formato libre por pantalla, el usuario podrá introducir el dato de manera libre. Esto significa que por ejemplo el valor 1.437 podrá introducirlo de muchas formas posibles, por ejemplo 1.437 ó 0.1437E+01.
PRINT*,A	• De igual forma en este caso el ordenador escribirá de manera libre el valor **A**

Por otro lado las sentencias READ(10,*) y WRITE(10,*) sin formato, para la lectura y escritura de un numero real por fichero tenemos que:

SENTENCIA	ACCION
READ(10,*)A	• Leerá una variable **A** con formato libre en un fichero UNIDAD=10, en el que el usuario habrá escrito el dato de manera libre.
WRITE(10,*)A	• De igual forma en este caso el ordenador escribirá de manera libre el valor **A** en un fichero UNIDAD=10.

El símbolo *, contenido entre los paréntesis, significa la ausencia de formato en la sentencia que lo contiene, este símbolo es sustituido por el número de etiqueta que contiene el formato establecido cuando se desea una ejecución con formato. A continuación se muestra la descripción de ambas sentencias:

• LECTURA CON FORMATO

READ(*Unidad, Etiqueta*)**Variable** *Etiqueta* **FORMAT**(*Código de formato*)	*Unidad*: Numero de unidad del fichero en el que se desea leer, o un *, para leer por pantalla. *Etiqueta:* Numero entero que indica la etiqueta en la que se especifica el formato de lectura. (**NOTA:** revisar el capítulo 2 para confirmar las columnas permitidas para la escritura del numero de etiqueta de la línea de sentencia FORMAT) *Código de formato*: Aquí se establecen de que forma se desea leer la variable

Ejemplo:

```
      READ(*,20)A
20    FORMAT(I2)
```

- **ESCRITURA CON FORMATO**

WRITE(*Unidad, Etiqueta*)**Variable** *Etiqueta* **FORMAT**(*Código de formato*)	*Unidad*: Numero de unidad del fichero en el que se desea escribir, o un *, para escribir en pantalla. *Etiqueta:* Numero entero que indica la etiqueta en la que se especifica el formato de lectura. (**NOTA:** revisar el capítulo 2 para confirmar las columnas reservadas para la escritura del numero de etiqueta de la línea de sentencia FORMAT) *Código de formato*: Aquí se establecen de que forma se desea escribir la variable

Ejemplo:

```
      WRITE(*,20)A
20    FORMAT(E9.3)
```

8.2.- CODIGOS DE FORMATO

Una vez conocida la estructura básica de la lectura y escritura con formato, ahora es preciso conocer de manera detallada los numerosos códigos de formato que FORTRAN permite para trabajar con los diferentes tipo de datos y su posicionamiento. En los dos ejemplos anteriores han sido incluidos dos códigos de formato **I2** y **E9.3** que ahora podrán ser comprendidos. Como ya hemos visto los códigos de formato pueden ser incluidos en la sentencia de entrada de datos READ o en la de salida WRITE, este hecho implica ciertas diferencias de comportamiento de los códigos que serán tratadas de manera individual. Por otro lado existen dos grandes grupo de códigos uno de ellos dedicado al tratamiento de datos y otro dedicado al posicionamiento de esos datos.

CODIGOS DE FORMATO PARA DATOS

FORMATO DE ENTEROS			
CODIGO	**TIPO DE DATO**	**FORMA**	**DESCRIPCION**
I	**ENTERO**	**Ia**	**I** : *Carácter de control que indica tratamiento de enteros.* **a** : *Entero que indica la longitud del dato numérico, debe incluirse signo.*

CODIGO **I** PARA **ENTRADA** DE DATOS:

Ejemplo:

```
      INTEGER A,B
      READ(*,10)A,B
10    FORMAT(I5,I4)
```

Si introducimos los siguientes datos:

```
BB576B-4B
```

Siendo B un espacio en blanco.

```
El resultado será: A=576 y B=-40
```

Obsérvese que:

- Los números han sido escritos de manera continua.
- La lectura ha respetado la longitud de cada uno de ellos establecida en el formato.
- El espacio en blanco final ha sido considerado como un cero y B ha tomado el valor -40

CODIGO **I** PARA **SALIDA** DE DATOS:

Ejemplo:

```
        INTEGER II,JJ,KK
        II=234
        JJ=12
        KK=-90
        WRITE(*,50)II,JJ,KK
 50     FORMAT(I4,I4,I4)
```

```
La salida será:

B234BB12B-90
```

Siendo B un espacio en blanco.

Como puede observarse los números han sido escritos de manera continua respetando la longitud de cada uno de ellos. Por otro lado para formatos repetitivos como el caso anterior, la sentencia FORMAT podría haberse sustituido por:

```
 50      FORMAT(3I4)
```

Lo que habría simplificado la sentencia manteniendo el resultado final. Esta simplificación también podrá aplicarse a los restantes formatos.

Existen por otra parte algunas consideraciones interesantes que pueden verse en el siguiente ejemplo:

Ejemplo:

```
        INTEGER S,C
        S=23234
        C=12
        WRITE(*,50)S
        WRITE(*,50)C
 50     FORMAT(I4)
```

```
La salida será:

****
BB12
```

Obsérvese que:

- Una misma sentencia de formato ha sido empleado por varias sentencia WRITE.

- El numero S es un entero de 5 dígitos, al ser el formato de escritura inferior en longitud, FORTRAN escribirá 4 asteriscos, significando la imposibilidad de la escritura bajo el formato establecido.

FORMATO DE REALES			
CODIGO	**TIPO DE DATO**	**FORMA**	**DESCRIPCION**
F	**REAL**	**Fa.b**	**F** : *Carácter de control que indica tratamiento de reales con coma flotante.* **a** : *Entero que indica la longitud total del dato numérico, debe incluirse signo y el espacio ocupado por la coma.* **b** : *Entero que indica el numero de cifras decimales que se desea*

CODIGO **F** PARA **ENTRADA** DE DATOS:

Ejemplo:

```
        REAL X,Y,Z
        READ(*,10)X,Y,Z
 10     FORMAT(F5.2,F6.1,F6.2)
```

```
ENTRADA:

B3.2B-1234B6789B
```

Siendo B un espacio en blanco.

```
VALOR:      X=3.2
            Y=-123.4
            Z=67.89
```

Obsérvese con detenimiento lo siguiente:

- Primeros 5 dígitos: `B3.2B`, lectura con formato F5.2 resultado X = 3.2
- 6 dígitos siguientes: `-1234B`, no existe punto decimal sin embargo al hacer la lectura con F6.1 FORTRAN toma el valor situado mas a la derecha como ultimo dígito decimal así pues Y = -123.4
- 5 dígitos siguientes: `6789B`, no existe tampoco punto decimal, luego Z = 67.89 para lectura con F6.2.

CODIGO **F** PARA **SALIDA** DE DATOS:

Ejemplo:

```
        REAL X,Y,Z
        X=5237
        Y=-177.1203
        Z=44.9999

        WRITE(*,10)X
        WRITE(*,20)Y,Z
 10     FORMAT(F9.3)
 20     FORMAT(F10.2,F6.2)
```

```
SALIDA:

BB5237.00
BBB-177.12B45.00
```

Siendo B un espacio en blanco.

Obsérvese con detenimiento lo siguiente:

- X carece de punto decimal, sin embargo al escribirlo con formato F9.3 se expresa como BB5237.000
- Al expresar el numero con un numero de dígitos decimales menor los que pose el dato se produce un redondeo al valor mas próximo. Y ha sido redondeado hacia abajo y Z hacia arriba.

FORMATO DE REALES			
CODIGO	**TIPO DE DATO**	**FORMA**	**DESCRIPCION**
E	**REAL**	**Ea.b**	**E** : *Carácter de control que indica tratamiento de reales con exponente.* **a** : *Entero que indica la longitud total del dato numérico, debe incluirse signo, el dígito que precede a la coma, el espacio ocupado por el punto decimal, la mantisa y el exponente.* **b** : *Entero que indica el numero de cifras dedicadas a la mantisa.*

CODIGO **E** PARA **ENTRADA** DE DATOS:

Ejemplo:

Supongamos que deseamos introducir por teclado las siguientes variables:

```
X=56.98765E02
Y=987.7654E-8
Z=0.00023E-1

5698765E02BB987.7654E-8BBB00023E-1
```

Siendo B un espacio en blanco.

Para realizar una lectura correcta se debería aplicar el siguiente formato:

```
        REAL X,Y,Z
        READ(*,10)X,Y,Z
 10     FORMAT(F10.5,F13.4,F11.5)
```

CODIGO **F** PARA **SALIDA** DE DATOS:

Ejemplo:

```
        REAL X,Y,Z
        X=5237098
        Y=-177.1203E-2
        Z=0.12345E+2
        WRITE(*,10)X
        WRITE(*,20)Y,Z
 10     FORMAT(E10.3)
 20     FORMAT(E10.2,E9.4)

SALIDA:

B0.524E+07
B-0.18E+07*********
```

Siendo B un espacio en blanco.

Obsérvese con detenimiento lo siguiente:

- Se produce un redondeo equivalente al que se producía para el formato F
- En caso de no ser posible al representación con el formato establecido FORTRAN rellena el campo presentado con asteriscos.

NOTA: Para el caso de estar trabajando en DOBLE PRECISION el formato **E** se debe sustituir por el formato **D** el cual sigue las mismas reglas que el **E**.

FORMATO PARA EL CONTROL DEL SIGNO (Únicamente para salidas)			
CODIGO	**TIPO DE DATO**	**FORMA**	**DESCRIPCION**
SP **SS** **S**	**REAL** **O** **ENTERO**	**SP** **SS** **S**	**SP,SS y S** : *Carácter de control que indica tratamiento del signo en la salida de reales o enteros.*

CODIGO **SP** UNICAMENTE PARA **SALIDA** DE DATOS:

En FORTRAN la presentación de datos numéricos se realiza omitiendo el signo positivo delante del primer dígito, para evitar esta carencia se utiliza el código SP.

Ejemplo:

```
        INTEGER  X
        REAL Y,Z
        X=3
        Y=34.56
        Z=56.78
        WRITE(*,10)X,Y,Z
 10     FORMAT(I1,SP,F6.2,F6.2)
```

```
SALIDA:

BB3+34.56+56.78
```

Siendo B un espacio en blanco.

Obsérvese con detenimiento lo siguiente:

- El efecto de la inclusión del signo positivo tiene efecto sobre las variables que se presentan con los formatos situados a la derecha del código SP.
- Si el formato es muy largo y contiene los códigos de muchas variables y se desea en algunas de ellas restablecer la opción de no incluir el signo, se debe incluir el código **SS**, con lo que las variables con formato situadas a la derecha ya no aparecerán con signo +. El restablecimiento de la inclusión del signo se activaría incluyendo el código S.

Ejemplo:

```
      WRITE(*,10)A,B,C,D,E
  10  FORMAT(F5.2,SP,F6.2,SS,F6.2, F6.2,S,F6.2)
```

```
Según lo anterior la variable A se presentara sin signo (situación por
defecto), B con signo, C y D sin signo y E con signo.
```

FORMATO PARA EL CONTRO DEL ESPACIOS EN BLANCO (Únicamente para entradas)			
CODIGO	**TIPO DE DATO**	**FORMA**	**DESCRIPCION**
BZ **BN**	**CUALQUIERA**	**BZ** **BN**	**BZ**: *Carácter de control que provoca interpretación de los espacios en blanco en la lectura de datos como ceros.* **BN:** *Carácter de control que hace ignorar los espacios en blanco en la lectura de datos.*

CODIGO **BZ** UNICAMENTE PARA **ENTRADA** DE DATOS:

En FORTRAN cuando se procede a la lectura de datos numéricos los espacios en blanco situados a la izquierda de un dato son ignorados y cuando un campo de entrada esta completamente en blanco FORTRAN interpreta esa situación como un cero. Sin embargo existen situaciones comprometidas como la existencia de un blanco entre dos dígitos o al final del dato que se ignoran pero que pueden ser interpretadas de modo diferente. Para ello están los códigos BZ y BN los cuales pueden alterar la interpretación que FORTRAN hace para ciertas situaciones de lectura de datos.

Ejemplo:

Supongamos que tenemos una lectura sobre la siguiente línea de información:

ENTRADA

BBBB45B7897B23

Siendo B un espacio en blanco.

CODIGO

```
        INTEGER  X,Y,Z
        READ(*,10)X,Y,Z
 10     FORMAT(I4,BZ,I6,BN,I4)
```

SALIDA:

```
X=0
Y=450789
Z=723
```

Obsérvese con detenimiento lo siguiente:

- El campo de X esta completamente vacío luego se interpreta como un cero que es la opción por defecto de FORTRAN.
- Y presenta un espacio en blanco intercalado, sin embargo su código de formato esta afectado por el código BZ luego el espacio se interpretara como un cero.
- Z también presenta un espacio en blanco sin embargo ahora su código de formato esta afectado por el código BN por lo que el espacio se omitirá.

FORMATO DE VARIABLE LOGICAS			
CODIGO	**TIPO DE DATO**	**FORMA**	**DESCRIPCION**

L	**LOGICO**	**La**	**L** : *Carácter de control que indica tratamiento de variables lógicas.* **a** : *Entero que indica la anchura del campo o lo que es lo mismo el numero de dígitos que ocupa el dato lógico.*

CODIGO **L** PARA **ENTRADA** DE DATOS:

Ejemplo:

VALOR DE ENTRADA	CODIGO DE LECTURA	VALOR ADQUIRIDO
BB·TRUE·	L9	.TRUE.
BB·TBBBB	L9	.TRUE.
BFB	L3	.FALSE.
BB·FBBB	L7	.FALSE.

Siendo B un espacio en blanco.

CODIGO **L** PARA **SALIDA** DE DATOS:

Ejemplo:

```
        LOGICAL X,Y
        X=.TRUE.
        Y=.FALSE.
        WRITE(*,10)X
        WRITE(*,20)X,Y
 10     FORMAT(L3)
 20     FORMAT(L4,L2)
```

```
SALIDA:

BBT
BBBTBF
```

Siendo B un espacio en blanco.

Obsérvese con detenimiento lo siguiente:

- La salida de una variable lógica siempre es T o F.
- El valor T o F ocupa siempre la posición mas a la derecha del conjunto de los digitados que completan el campo definido por el formato.

FORMATO DE VARIABLES CARACTER			
CODIGO	**TIPO DE DATO**	**FORMA**	**DESCRIPCION**
A	**CARACTER**	**Aa**	**A** : *Carácter de control que indica tratamiento de caracteres.* **a** : *Entero que indica la longitud total del dato tipo carácter.*

CODIGO **A** PARA **ENTRADA** DE DATOS:

Ejemplo:

```
ENTRADA:

HOLADESTORNILLADOR
```

Para realizar una lectura correcta se debería aplicar el siguiente formato:

```
        CHARACTER*4 X
        CHARACTER*6 Y
```

```
        READ(*,10)X,Y
 10     FORMAT(A4,A6)

VALOR:

X='HOLA'
Y='DESTOR'
```

Obsérvese con detenimiento lo siguiente:

- La entrada no ha requerido introducir los caracteres entrecomillados como se ha hecho hasta ahora, esta es una gran diferencia con respecto a la entrada de caracteres bajo formato libre.
- En caso de leer con formato de longitud inferior a la longitud real del dato, se almacenaran las posiciones establecidas por el formato empezando por al izquierda.

CODIGO **A** PARA **SALIDA** DE DATOS:

Ejemplo:

```
        CHARACTER*4  X
        CHARACTER*17 Y
        CHARACTER*7  Z
        X='HOLA'
        Y='ESTOY APRENDIENDO'
        Z='FORTRAN'
        WRITE(*,10)X
        WRITE(*,20)Y
        WRITE(*,30)Z
 10     FORMAT(A4)
 11     FORMAT(A10)
 20     FORMAT(A12)

SALIDA:

HOLA
ESTOY APRE
BBBBBFORTRAN
```

Obsérvese con detenimiento lo siguiente:

- En caso de que el formato establecido en FORTRAN tenga una longitud superior al de la variable se rellenará el campo sobrante empezando por la izquierda con espacios en blanco.

FORMATO DE VARIABLES CARACTER
(Únicamente para salidas)

CODIGO	TIPO DE DATO	FORMA	DESCRIPCION
H	**CARACTER**	**aH**	**H** : *Carácter de control que indica tratamiento de caracteres.* **a** : *Entero que indica la longitud total del dato tipo carácter.*

CODIGO **H** UNICAMENTE PARA **SALIDA** DE DATOS:

Ejemplo:

En ocasiones hemos empleado la siguiente construcción para producir una salida de tipo carácter:

```
        CHARACTER*18 Y
        Y='ESTOY EN SANTANDER'
```

```
        WRITE(*,10)Y
 10     FORMAT(A18)

SALIDA:

ESTOY EN SANTANDER
```

En caso de emplear el código H, el código anterior se transformaría en:

```
        WRITE(*,10)
  10    FORMAT(18HESTOY EN SANTANDER)
```

Obsérvese con detenimiento lo siguiente:

- En caso de emplear el código H no es preciso declarar variable alguna.
- El mensaje ESTOY EN SANTANDER ocupa 18 espacios de ahí que se haya indicado 18H.
- En caso de haber deseado incluir tres espacios en blanco al final del mensaje el formato debería haberse construido de la siguiente forma: `FORMAT(21HESTOY EN SANTANDERBBB).` Siendo `B` un espacio en blanco.

CODIGOS DE FORMATO DE POSICIONAMIENTO

Con los formatos anteriores se ha resuelto el problema de la presentación y adquisición de datos de cualquier tipo bajo un formato establecido, sin embargo los datos deben estar ordenados por columnas, separados por varios espacios o tabuladores y situados en diferentes filas según lo requiera una mínima comprensión de la información, este problema es ya de posicionamiento del dato. A continuación ser presentarán este tipo de formatos:

En este tipo de formatos no es preciso diferenciar su comportamiento cuando se emplean en la sentencia READ y en la sentencia WRITE puesto que su función no varia.

FORMATO PARA EL CONTROL DE ESPACIOS		
CODIGO	**FORMA**	**DESCRIPCION**
X	**aX**	**X** : *Carácter de control que indica tratamiento de espacios el línea.* **a** : *Entero que indica numero de espacios que han de incluirse.*

Ejemplo:

```
ENTRADA:

BBB12.34BBBBB567BBBB8.98
```

Para realizar una lectura correcta se debería aplicar el siguiente formato:

```
        INTEGER B
        REAL    A,C
        READ(*,10)A,B,C
  10    FORMAT(3X,F5.2,5X,I3,4X,F4.2)

VALOR:

A=12.34
Y=567
Z=8.98
```

FORMATO PARA EL CONTROL DE TABULADORES

CODIGO	FORMA	DESCRIPCION
T	**Ta**	**T** : *Carácter de control que indica tratamiento de tabuladores.* **a** : *Entero que indica la longitud del tabulador.* *NOTA: Por ejemplo T11 equivale a 10X. Ambas significan que han de saltarse 10 posiciones y debe iniciarse la acción en la posición 11.*

Ejemplo:

```
        REAL    A,B,C
        A=12.34
        Y=34.56
        Z=45.78
        WRITE(*,10)A,B,C
  11    FORMAT(3(T5,F5.2))
```

SALIDA:

BBBB12.34BBBB34.56BBBB45.78

FORMATO PARA EL CONTROL DE FIN DE REGISTRO		
CODIGO	**FORMA**	**DESCRIPCION**
/	/	/ : *Carácter de control que indica que se deberá saltar al siguiente registro o lo que es lo mismo se deberá saltar a la siguiente línea*

Ejemplo:

Supongamos que tenemos en un fichero (unit=10) los siguientes datos:

BBBB12.34BBBB34.56
BBBB6456.56

La cual podría haberse generado con el siguiente código y formato:

```
        REAL    A,B,C
        A=12.34
        B=34.56
        C=6456.56
        WRITE(10,20)A,B
        WRITE(10,21)C
   20   FORMAT(2(T5,F5.2))
   21   FORMAT(T6,F7.2)
```

Pues bien, para abordar su lectura incluyendo el código de formato /, se establece el siguiente código:

```
        REAL    A,B,C
        READ(10,20)A,B,C
   20   FORMAT(2(T5,F5.2),/,T6,F7.2)
```

Obsérvese con detenimiento lo siguiente:

- La inclusión del código / nos permite saltar a la siguiente línea sin tener que incluir una nueva sentencia READ.
- Un formato tal que: `FORMAT(///),` saltará o generará 3 líneas en blanco según sea empleada para la lectura o la escritura.

8.1.- RECOMENDACIONES PARA LA CREACIÓN DE FICHEROS

- La lectura de datos a través de un fichero de entrada debe hacerse SIEMPRE mediante lectura formateada.

- Todos los datos de los ficheros de entrada deben estar perfectamente documentados en el propio fichero, incluyendo, si lo tienen, sus dimensiones. Por ejemplo, si hay que introducir un dato numérico correspondiente a la frecuencia de funcionamiento en Gigahercios de un cierto elemento, se podría escribir un fichero de entrada con esta forma

```
Frecuencia de funcionamiento (Ghz)
**.****
12.3550
```

  que habría que leer de la forma adecuada mediante las instrucciones antes relacionadas. Con los asteriscos se indica la posición exacta donde escribir las cifras del dato correspondiente, en este caso la frecuencia. Se puede idear cualquier otra manera que sea igualmente clara.

- En el fichero de salida deben incluirse SIEMPRE los datos de entrada en la forma que se considere más adecuada. Ésto permite conocer la respuesta a las entradas sin necesidad de examinar diferentes ficheros.

- Hay que tener en cuenta que un programa está terminado cuando el código está perfectamente comentado y los datos y resultados están debidamente presentados. Para la realización de estas dos labores, especialmente de la segunda, que llamamos depuración del programa, es habitual emplear entre el 50 y el 70% del tiempo total empleado en la realización de un programa. Es por ello que la paciencia, en la realización de estas tareas en cierto modo rutinarias, es la característica principal para poder programar con la suficiente solvencia.

EJERCICIO 8.1

- Modificar el programa del ejemplo 7.3, para leer N (como máximo 20) números enteros con formato I (el rango puede ser de –999 a 999) de un fichero de entrada, ordenarlos de mayor a menor y escribirlos en un fichero de salida con un formato I.

EJERCICIO 8.2

- Rehacer el programa de la ecuación de 2º grado, leyendo los coeficientes de un fichero con formato F7.2. Escribir el resultado en un fichero de salida especificando los tipos de raíces con el formato F que se considere adecuado. Repetir el programa escribiendo la salida con formato E. Recordar que el fichero de salida debe incluir los datos de entrada.

- Modificar el fichero de entrada especificando el significado de cada dato de entrada en el sentido indicado en la nota anterior. Rehacer el programa para que sea capaz de leer ese fichero.

EJERCICIO 8.3

- Rehacer el programa del ejercicio 5.3 leyendo y escribiendo, con formato, los datos en ficheros con los comentarios correspondientes. Realizar un programa leyendo con formato F y escribiendo con formato F y después realizar otro programa escribiendo con formato E. Por último, realizar otro programa que sea capaz de leer del fichero de salida los datos numéricos (es decir, los valores de los argumentos y los valores de la función) y escribirlos en otro fichero.

- Nota: en los ejemplos anteriores utilizar el máximo posible de comentarios para practicar con los diferentes formatos explicados en este capítulo.