Ficheros

Introducción

Podemos definir un fichero o archivo como una unidad de información almacenada en memoria secundaria, un disco, a la que se asigna un identificador único; los ficheros nos ofrecen la posibilidad de almacenar datos de una manera permanente.

Los ficheros pueden considerarse como estructuras secuenciales en las que se almacenan datos pertenecientes a un mismo tipo; así podemos tener ficheros de texto (caracteres), ficheros de tipo entero o, más habitualmente, ficheros de registros (por ejemplo, un fichero que almacene los datos de los alumnos de una asignatura).

Esta estructuración de los ficheros da lugar a dos formas distintas de acceso a los mismos: secuencial y directo. En el primer caso, para leer o escribir un registro en el fichero es necesario pasar por todos los registros previos (de manera análoga a como se trabajaría con una cinta); en el caso del acceso directo, cada registro tiene asociada una clave que permite un acceso inmediato sin tener que recorrer previamente otros registros.

La mayor parte de los lenguajes de programación, incluido FORTRAN, permiten ambos tipos de acceso, sin embargo, el más habitual es el acceso secuencial por lo que sólo estudiaremos ese tipo.

Como ya hemos dicho los ficheros pueden accederse de forma secuencial o directa; en el caso de los accesos secuenciales para leer o escribir un dato dado en el fichero es necesario leer o escribir todos los datos previos recorriendo el fichero elemento a elemento (este elemento puede ser un entero, un carácter, o un registro en función del tipo de fichero). La analogía que se suele emplear para describir esta forma de acceso a ficheros es la de una cinta puesto que para acceder a un punto de la misma es necesario recorrer el tramo que va desde el inicio de la cinta al punto deseado.

Existe otra forma de acceder a la información que es el acceso directo, sin embargo, el acceso secuencial es mucho más frecuente razón por la cual en nuestra notación algorítmica supondremos que la única forma de acceder a los ficheros es de forma secuencial, esto es, elemento a elemento.

Una vez aclarado este punto diremos que en la notación algorítmica los ficheros son un tipo de dato construido a partir de un único tipo base; así, podemos tener ficheros de enteros, de reales, de caracteres (ficheros de texto) o ficheros de tuplas (registros). Un paso previo, por tanto, para poder trabajar con ficheros será declarar una o más variables fichero del tipo adecuado.

La sintaxis empleada para declarar una variable de tipo fichero es la siguiente:

```
variable ∈ fichero de tipo
```

A continuación se muestran algunos ejemplos:

```
numeros ∈ fichero de entero
carta ∈ fichero de caracter
matriculacion ∈ fichero de alumno
```

El tipo fichero requiere para su empleo una serie de acciones que se implementan como procedimientos; tales acciones son las siguientes: apertura, cierre, lectura, escritura y fin de fichero.

Apertura y cierre de ficheros

El hecho de disponer de una variable de tipo fichero no significa nada, si deseamos leer datos del fichero o escribir datos en el mismo es necesario “abrirlo”; esta acción prepara el fichero para poder llevar a cabo el resto de acciones. Podemos imaginar que por cada fichero existe una “cabeza de lectura/escritura”, dicha cabeza se coloca siempre al comienzo del fichero cuando éste es abierto.

La operación simétrica es el cierre del fichero, una vez se han llevado a cabo todas las operaciones con un fichero determinado es necesario cerrarlo; en caso de que se hayan almacenado datos en el fichero estos se grabarían finalmente en el momento del cierre.

La sintaxis de ambas operaciones es la siguiente:

```
abrir (fichero)
cerrar (fichero)
```

Una peculiaridad de los ficheros en esta notación algorítmica respecto a los ficheros empleados en los lenguajes de programación es que en ningún momento se indica la ruta del fichero, es decir, la unidad, directorio y nombre del fichero “físico” (por ejemplo, `a:\carta.txt` o `c:\mis documentos\apuntes.doc`); la razón de esta peculiaridad es muy simple: este pseudolenguaje sólo pretende servir para especificar de una manera más o menos detallada y no ambigua aspectos importantes previos a la codificación en un lenguaje de programación, creemos que indicar la ruta de los ficheros en el momento de diseñar el algoritmo no es un aspecto esencial y, por tanto, prescindimos de las mismas.

A continuación se muestran unos ejemplos sencillos:

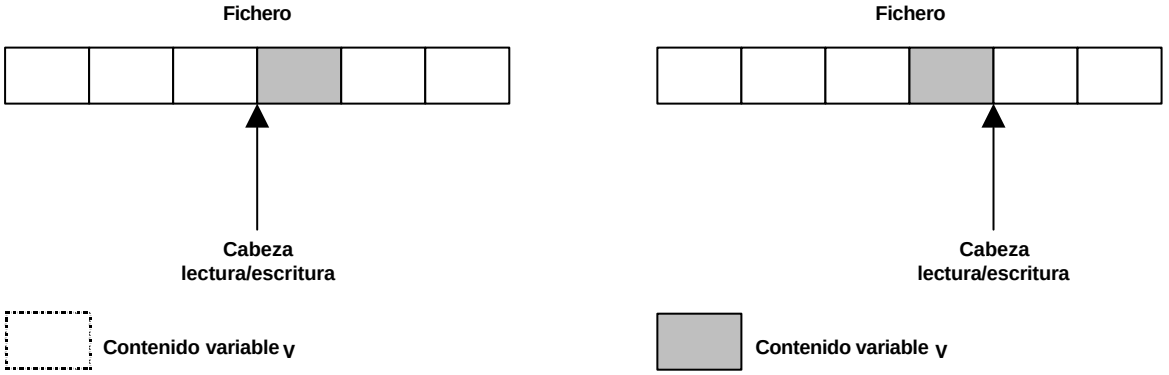
```
variables
  numeros ∈ fichero de entero
  carta ∈ fichero de carácter

inicio
  abrir (numeros)
  abrir (carta)

  ...

  cerrar (numeros)
  cerrar (carta)
fin
```

Lectura/escritura de ficheros

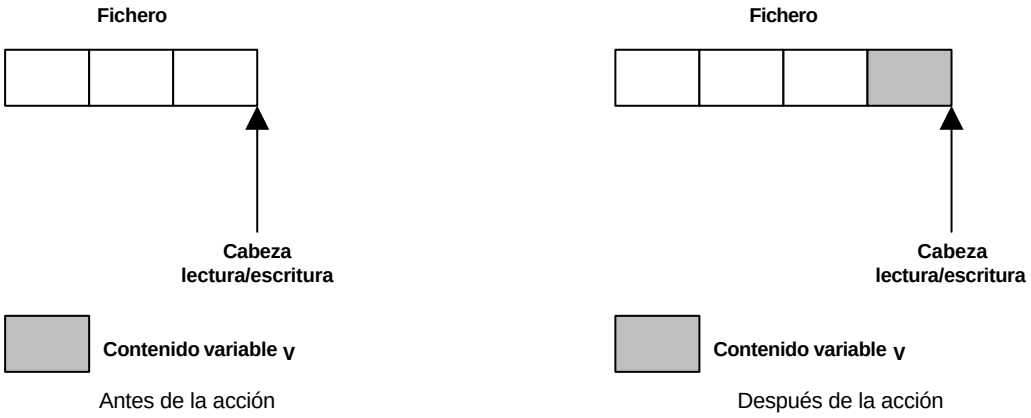
Una vez el fichero ha sido abierto es posible escribir o leer datos del mismo siendo estos datos del tipo base del fichero abierto. Al realizarse una operación de lectura/escritura sobre un fichero la “cabeza de lectura/escritura” avanza siempre una posición. La sintaxis de ambas operaciones es la siguiente:

```
leer (fichero, variable)
escribir (fichero, expresión)
```

La acción del procedimiento `leer(fichero,v)` se comporta como se muestra a continuación:



Mientras que la acción `escribir(fichero,v)` se comporta de la manera siguiente:



Fin de fichero

Al leer un fichero es necesario saber si se ha llegado al final del mismo; existe una variable lógica denominada `fin_fichero` que retorna falso si la cabeza de lectura/escritura aún no ha alcanzado el final del fichero y verdadero en caso contrario. La sintaxis de la misma es la siguiente:

```
fin_fichero (fichero)
```

Ejemplos del uso de ficheros en la notación algorítmica

A continuación se mostrarán dos sencillos ejemplos del uso de ficheros; el primer ejemplo muestra un algoritmo que lee un número indeterminado de enteros mayores o iguales que 0 y los almacena en un fichero, el segundo recorre un fichero con datos personales mostrándolos por pantalla.

```
variables
  numeros ∈ fichero de entero
  numero ∈ entero

inicio
  escribir 'introduzca enteros positivos:'

  numero ← 0

  abrir (numeros)

  mientras numero ≥0 hacer
    leer numero

    si numero ≥0 entonces
      escribir (numeros, numero)
    fin si
  fin mientras

  cerrar (numeros)
fin
```

Notas:

- El algoritmo debe utilizar una variable de tipo entero para leer los números por teclado y una variable de tipo fichero de enteros para almacenar en dicho fichero los números.
- Al principio del algoritmo el fichero debe abrirse.
- Al final del algoritmo el fichero debe cerrarse.
- El algoritmo se basará en un bucle que leerá números mientras dichos números no sean negativos.
- Si el número leído es mayor o igual que 0 lo almacena en el fichero.

```
tipos
  alumno = tupla
    nombre, apellidos ∈ caracter
  fin tupla

variables
  datos_alumno ∈ alumno
  matriculacion ∈ fichero de alumno

inicio
  abrir (matriculacion)

  mientras no fin_fichero (matriculacion) hacer
    leer (matriculacion, datos_alumno)

    escribir 'nombre: ', datos_alumno.nombre
    escribir 'apellidos: ', datos_alumno.apellidos
  fin mientras

  cerrar (matriculacion)
fin
```

Notas:

- El algoritmo debe utilizar una variable de tipo fichero de registros de alumnos que almacena los datos (aunque este algoritmo no ha escrito el fichero) y una variable de tipo registro para leer desde el fichero y mostrarlo por pantalla.
- Al principio del algoritmo el fichero debe abrirse.
- Al final del algoritmo el fichero debe cerrarse.
- El algoritmo se basará en un bucle que leerá registros del fichero mientras no se llegue al final del mismo; el contenido de cada registro leído se mostrará por pantalla.

Uso de ficheros en FORTRAN

FORTRAN como todos los lenguajes de programación modernos permite el manejo de ficheros; ya en la introducción mencionamos que era posible utilizar tanto ficheros de acceso secuencial como directo aunque nosotros sólo utilizaríamos los primeros. Sin embargo, como en tantas otras cosas, FORTRAN también es especial a la hora de manejar ficheros puesto que no dispone de ningún tipo fichero.

Para acceder a los ficheros el lenguaje FORTRAN utiliza lo que se denominan números de unidad; a cada fichero abierto en un programa se le debe asignar un número de unidad, variando dichos números en el rango [1, 99] y existiendo dos números de unidad reservados: el 5 y el 6, correspondiéndose al teclado y a la pantalla, respectivamente.

Por lo que respecta a las operaciones necesarias para trabajar con ficheros son las mismas, aún no existiendo un tipo fichero, y difieren tan sólo en la sintaxis.

Apertura y cierre de ficheros

Las operaciones de apertura y cierre de ficheros en FORTRAN se comportan de manera muy semejante a sus homólogas en la notación algorítmica, diferenciándose tan sólo en la sintaxis a emplear:

```
open (unit=unidad, file=ruta del fichero, status='new'|'old'|'unknown')

close (unidad)
```

Como se puede ver, la acción open recibe tres argumentos:

- **Unidad:** Un número entero entre 1 y 99, ficheros distintos deben tener asignados números de unidad diferentes y no se debe abrir un fichero sobre una unidad ya asignada sin cerrar previamente el fichero antiguo. Si se abre de esta forma la unidad 5 podríamos leer del teclado (aunque nunca escribir) y si es la unidad 6 la abierta podríamos escribir en pantalla (aunque nunca leer). El número de unidad asignado a un fichero en el momento de la apertura es fundamental puesto que será utilizado a partir de ese momento en todas las operaciones que deban acceder al fichero.

- **Ruta del fichero:** Un fichero siempre se almacena en memoria secundaria, esto es, en disco; mientras que en la notación algorítmica la ruta del fichero nos era indiferente resulta muy importante a la hora de implementar un programa en un lenguaje. La ruta es una cadena de caracteres (un literal o una variable) que indica la "localización" del fichero, por ejemplo: , `'documento.doc'` o `'c:\dir\subdir\alum.dat'`.
- **Estado del fichero:** En nuestra notación algorítmica no prestábamos atención al estado de los ficheros, esto es, si no existían y era necesario crearlos o si ya existían previamente; en FORTRAN es posible indicar tres estados diferentes:
    - `'new'`: El fichero no existe y es necesario crearlo, lógicamente este estado sólo debe utilizarse cuando se va a crear el fichero nunca para leerlo o sobreescribirlo.
    - `'old'`: El fichero ya existe, este estado sólo debería utilizarse cuando el fichero se va a leer o sobreescribir pero nunca para crearlo.
    - `'unknown'`: Un estado "comodín", si el fichero no existiera se comportaría como y si existiera como `'old'`.

A continuación se muestran algunos ejemplos sencillos:

```
! Se abre un fichero que no existe (para crearlo)
!
open (unit=1, file='carta.txt', status='new')
...
close (1)
...
! Se abre un fichero que ya existe (para leerlo o sobreescribirlo)
!
open (unit=2, file='matriculacion.dat', status='old');
...
close (2)
```

Lectura/escritura de ficheros

Las acciones de lectura son también muy similares aunque la sintaxis cambia un poco:

```
read (unidad, formato) variable
write (unidad, formato) expresión
```

`unidad` es el número asignado en la sentencia de apertura mientras que `formato` es un código de formato como los vistos en la tercera lección (Declaración de variables. Sentencias de entrada/salida); el mismo consejo que se dio entonces se repite aquí: utilizar siempre que sea posible el código de formato libre `*`.

Veamos algunos ejemplos sencillos:

```
open (unit=1,file='matriculacion.dat',status='old')
read (1,*) datos_alumno
close (1)
...
open (unit=2,file='carta.txt',status='old')
read (2,*) letra
close (2)
...
open (unit=1,file='matriculacion.dat',status='old')
write (1,*) datos_alumno
close (1)
```

Por lo que respecta al comportamiento de la "cabeza de lectura/escritura" es exactamente igual en FORTRAN

Fin de fichero

En FORTRAN también existe una función que permite determinar si se ha alcanzado el final de un fichero; la sintaxis es la siguiente:

```
eof (unidad)
```

Ejemplos del uso de ficheros en FORTRAN

Para terminar la lección se presentarán los ejemplos anteriormente descritos mediante la notación algorítmica traducidos al lenguaje FORTRAN:

```
program programa
  implicit none
  integer numero

  print *, 'Introduzca enteros positivos:'
  numero = 0

  open
      (unit=1,file='numeros.dat',status='unknown')

  do while (numero>=0)
    read *, numero
      if (numero>=0) then
        write (1,*) numero
      end if
  end do

  close (1)
end
```

```
program programa
  implicit none

  type alumno
    character*64 nombre
    character*128 apellidos
  end type alumno

  type (alumno) datos_alumno

  open (unit=1,file='matriculacion.dat',status='unknown')

  do while (.not.eof(1))
    read (1,*) datos_alumno
      print *,'Nombre: ',datos_alumno%nombre,'Apellidos:
      ',datos_alumno%apellidos
  end do

  close (1)
end
```

Resumen

1. Un fichero es una unidad de información almacenada en disco a la que se asigna un identificador único.
2. Los ficheros son estructuras secuenciales en las que se almacenan datos pertenecientes a un mismo tipo.
3. Hay dos formas de acceso a ficheros: secuencial y directo.
4. En la notación algorítmica sólo existen ficheros secuenciales y para utilizar un fichero es necesario declararlo.
5. En FORTRAN aunque existen ficheros de acceso directo no los utilizaremos; además, no existe un tipo fichero sino

6. Las operaciones básicas con un fichero son:
   - Apertura.
   - Cierre.
   - Lectura/Escritura.
   - Fin de fichero.