## Lectura de datos en formato binario

### Función FREAD

La función: fread, se encarga de leer un número determinado de elementos y guardarlos en un determinado buffer. Además, el indicador de posición del fichero para su estructura FILE asociada lo avanzará el número de elementos leídos correctamente; es decir, que moverá el cursor en consecuencia de los datos leídos correctamente.

El prototipo de esta función es:

*size\_t fread(void \*buffer, size\_t size, size\_t count, FILE\* stream)*

Los parámetros que se le pasan a esta función son:

* *buffer*: puntero al buffer donde se almacenarán los elementos leídos del fichero.
* *size*: tamaño de cada uno de los elementos que se leerán del fichero.
* *count*: número de elementos que se leerán del fichero.
* *stream*: puntero a la estructura FILE asociada al fichero que queremos leer.

La función devolverá el número de elementos que se han leído correctamente, que puede ser menor que el número de elementos especificado: *count*, si se produce un error o si se llega al final del archivo. Si el valor de: *size* o *count*, es 0, entonces devolverá 0 y el contenido de: *buffer*, y el estado de: *stream*, permanecerán sin cambios.

## Lectura de datos en formato texto

### Función FGETC

La función: fgetc, se encarga de leer un carácter de un determinado buffer. Además, el indicador de posición del fichero para su estructura FILE asociada se avanzará al siguiente carácter a leer.

Su prototipo de función es:

*int fgetc(FILE\* stream)*

El parámetro que se le pasa a esta función es: *stream*, que es un puntero a la estructura FILE asociada al fichero del cual queremos leer el carácter.

La función devolverá el valor del carácter leído casteado a un dato int; es decir, que el carácter se leerá como un dato int.

#### Función FGETS

La función: fgets, se encarga de leer un número determinado de caracteres hasta que se lee el número de caracteres especificado, se obtiene un carácter de una nueva línea o se llega al final del fichero. Además, esta función mueve el cursor hasta el final de la línea leída.

El prototipo de esta función es:

*char \*fgets(char\* string, int n, FILE\* stream)*

Los parámetros que se le pasan a esta función son:

* *string*: puntero al buffer de la cadena de caracteres donde se almacenará los caracteres leídos. Si se llega al final de una línea, el carácter: \n, que indica el final de una línea también se almacenará.
* *n*: número máximo de caracteres a leer.
* *stream*: puntero a la estructura FILE asociada al fichero que queremos leer.

La función devolverá un puntero a la cadena: *string*, si la lectura ha sido exitosa. No obstante, si se alcanza el final del fichero y todavía no se había leído ningún carácter, entonces se devolverá NULL y la cadena: *string*, permanecerá invariable. Además, si ocurre un error durante el proceso, el contenido de la cadena: *string*, será indeterminado y se devolverá NULL.

#### Función FSCANF

La función: fscanf, se encarga de leer datos de un fichero, formateándolos según un formato específico. Es decir, la función leerá los datos del fichero e intentará buscar coincidencias con el formato especificado. Esta función ignorará los espacios en blanco (espacios, saltos de línea y tabulaciones) excepto si se usa: %c, que sí los considera. Además, moverá el cursor según lo leído; es decir, que moverá el cursor hasta que se paré de leer o exista un error.

El prototipo de esta función es:

*int fscanf(FILE\* stream, const char\* format\_string, argument\_list)*

Los parámetros que se le pasan a esta función son:

* *stream*: puntero a la estructura FILE asociada al fichero que queremos leer.
* *format*: puntero a la cadena de caracteres con el formato que define como se deben interpretar los datos leídos. Puede contener especificadores de formato como: %d, %s o %f.
* *argument\_list*: punteros a las variables donde se almacenarán los valores leídos.

La función devolverá el número de datos de entrada correctamente leídos y asignados. Este valor puede ser menor que el que se ha especificado en el formato de cadena, incluso cero, si hay un error en la asignación. Si antes de que ocurra cualquier conversión hay un error, se devuelve el valor EOF.