ILERNA Online

Videotutoría 10 (UF3): Ficheros

Módulo 03A: Programación

Conceptos

- Todo lo que llevamos visto hasta aquí ha sido mediante variables, estructuras de datos, y hemos manipulado la información de la que disponíamos.

- Esta información, una vez que finaliza la ejecución del software, desaparece de memoria, ya que ha estado almacenada en la memoria principal el tiempo que dura la ejecución del software.

- Podemos ver los **ficheros** como una parte de un **dispositivo no volátil** a la que se le asigna un nombre, y que puede contener una cantidad de datos que va a estar limitada, o por la cantidad de espacio del que disponga el dispositivo o por las características del sistema operativo.

- **Los ficheros o archivos son una secuencia de bits (0 y 1)** que se almacenan en un dispositivo de almacenamiento secundario, por lo que la información va a permanecer a pesar de que se cierre la aplicación que los utilice.

Conceptos

Esto da más independencia sobre la información, ya que no necesitamos que el programa se esté ejecutando para que la información que contiene exista

A la hora de trabajar con ficheros debemos tener en cuenta:

- La información es un conjunto de 0 y 1.
- Al agrupar los bits se forman bytes o palabras.
- Al agrupar los campos se crean los registros de información.
- Un fichero es creado por un conjunto de ficheros de manera que todos tienen en común la misma estructura.
- Los directorios tienen la función de agrupar distintos ficheros siguiendo unas condiciones determinadas dadas por el sistema operativo o por el programador.

Ruta de ficheros

Dependiendo de cómo empecemos la ruta de directorio para nombrar el archivo, podemos tener dos tipos de rutas bien diferenciadas:

c:\Ilerna\porjects\nombre.cs (en windows)

- ***Ruta absoluta o completa**: se le indica el camino de directorio desde el comienzo. Si es en el sistema operativo Linux o, empezará por la raíz (/), en caso contrario si el sistema operativo es Windows debe de empezar por el nombre de la unidad en cuestión.*

- ***Ruta relativa**: se le indica el camino de directorio desde la posición actual.*

XML es un lenguaje orientado a etiquetas, muy similar a HTML

Se abre com < ejemplo_dato >

ILERNA Online

Tipo de ficheros: secuencial

- ***Secuencial:*** en los ficheros secuenciales los registros se van almacenando en posiciones consecutivas de manera que cada vez que queramos acceder a ellos tendremos que empezar desde el primero e ir recorriéndolos de uno en uno.

- *Solo se puede realizar una operación de lectura o escritura a la vez. Por ejemplo, si estamos leyendo el fichero, no podemos realizar ninguna operación de escritura sobre él hasta que termine de ser leído y viceversa.*

Tipo de ficheros: Aleatorio

Este no lo usaremos.

- ***Aleatorio o directo***: *en los ficheros aleatorios o directos podemos acceder a un registro concreto del mismo indicando una posición perteneciente a un conjunto de posiciones posibles.*

Hay que programar el puntero. Este sistema está anticuado.

- *Debido a que los registros están organizados, estos pueden ser leídos o escritos en cualquier orden, ya que se accede a cada uno a través de su posición. Cuando queremos realizar una operación, basta con colocar el puntero que maneja el fichero justo antes de éste*

ILERNA Online

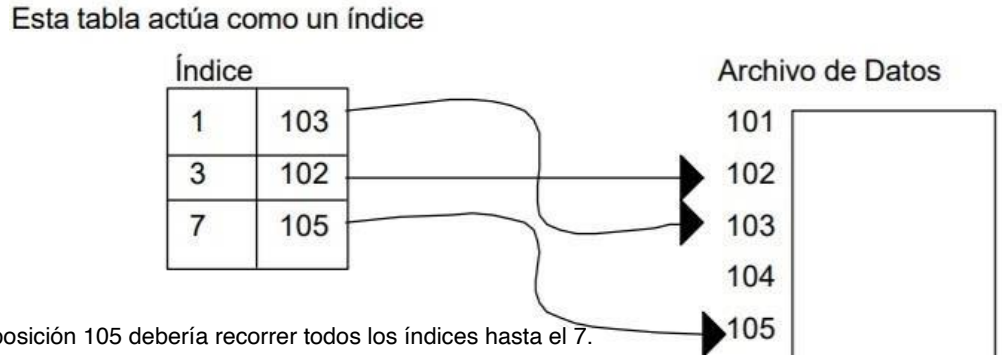
Tipo de ficheros: secuencial indexado

- ***Secuencial indexado***: los ficheros indexados poseen un campo clave para ser identificados.
- Permiten el acceso secuencial y aleatorios a un fichero de la siguiente forma:

- 1) Primero busca el índice en una tabla.
- 2)Una vez que lo encuentra, el acceso al fichero es directo, ya que solo tenemos que acceder a la posición indicada por el índice.

Tipo de ficheros: secuencial indexado

- Se utiliza este tipo de organización de archivo cuando existe la necesidad tanto de crear un acceso a los registros secuencialmente, por algún valor de llave, como de acceder a ellos individualmente.
- Un archivo secuencial indexado proporciona la combinación de tipos de acceso que manejan un archivo secuencial y un archivo relativo o de acceso directo.



Si quiero escribir o leer en la posición 105 debería recorrer todos los índices hasta el 7.

Estructura de ficheros

Archivo = fichero

- ***Archivo binario:*** *Es un archivo informático que contiene información de cualquier tipo codificada en binario para el propósito de almacenamiento y procesamiento en ordenadores.* Solo entendible por la máquina.

- *Los archivos binarios que contienen bytes suelen ser interpretados como alguna cosa que no sean caracteres de texto. Un ejemplo típico son los programas de ordenador compilados; de hecho, las aplicaciones o programas compilados son conocidos como binarios*

- ***Archivos de texto***

Operaciones sobre ficheros

- Lo primero que debemos hacer siempre es abrir el fichero para la operación que vayamos a realizar sobre él. Cuando abrimos el fichero estamos relacionando un objeto de nuestro programa con un archivo.

- Los diferentes modos en los que se puede abrir un fichero son los siguientes: Primero debemos comprobar si ya existe un fichero, y si ya esta escrito o si bien está vacío.
  - ***Lectura***: solamente va a realizar operaciones de lectura en el fichero.
  - ***Escritura***: realiza operaciones de escritura en el fichero. Si ya existía, lo sobrescribe.
  - ***Añadir***: permite realizar la operación de escritura en un fichero que ya existía añadiéndole a lo anterior

Lectura Secuencial

Mientras no lleguemos al final del fichero debemos ir leyendo registro a registro.
Una vez leído todos esos registros hacemos todas esas acciones que queramos.

```
//Declaración de la variable del fichero
fichero f1;

//Abrimos el fichero para leerlo
f1.abrir(lectura);

Mientras no final de fichero hacer
        f1. leer(registro);
        operaciones con registro leído;
finMientras

//Cerramos el fichero
f1.cerrar();
```

Se puede hacer por consola o bien escribir esa información en un fichero a la hora de guardarlo como tal.

Escritura Secuencial

```
//Declaración de la variable del fichero
fichero f1;

//Abrimos el fichero para leerlo
f1.abrir(escritura);

Mientras tengo información el fichro hacer
        Configurar registros a partir de unos datos
        f1. escribir(registro);  Escribir nuestros datos dentro del fichero.

finMientras

//Cerramos el fichero
f1.cerrar();
```

Lectura Aleatoria

```
// Declaración de la variable del fichero

fichero f1;


// Abrimos el fichero para leerlo

f1.abrir(lectura);
Mientras condición del programa Hacer
        Situar el puntero del fichero justo antes del registro requerido
        f1.leer(registro);
        operaciones con registro leído;
FinMientras


// Cerramos el fichero

f1.cerrar();
```

seelx para buscar nuestro registro

Una vez encontrado el registro que queramos y colocado el puntero ya realizamos las operaciones que nosotros queramos.

Escritura Aleatoria

```
// Declaración de la variable del fichero
fichero f1;

// Abrimos el fichero para leerlo
f1.abrir(escritura);

Mientras se deseen escribir datos Hacer
    Posicionar el puntero del fichero en la posición deseada
f1.escribir(registro);
operaciones con registro leído;
finMientras

//Cerramos el fichero
f1.cerrar();
```

Podemos hacer un array solo que en lugar de ejecutarlo por consola almacenamos sus datos.

Flujo de ficheros

Como un puente para poder guardar ese contenido que nosotros hemos trabajado a nuestro fichero físico. Pasar de la programación a lo físico.

- ***Los flujos (también llamado Stream) de datos son las estructuras o pasarelas que tenemos para acceder a los datos de un fichero, de una forma consistente y fiable, desde un código fuente en un cualquier lenguaje de programación.***

- En el flujo de Entrada de datos (**Lectura**) solo podemos realizar la operación de lectura de un fichero, es decir, existe una comunicación unilateral desde el fichero al programa.
- En el flujo de Salida (**Escritura**) también es en sentido unidireccional, ya que solo podemos realizar la operación de escritura en el fichero.
- El tipo de stream lo vamos a **definir al principio** de trabajar con ficheros y no se podrá cambiar una vez abierto.

Clase FileStream

- ***Proporciona un Stream para un archivo, lo que permite operaciones de lectura y escritura sincrónica y asincrónica.***
- Detallamos el constructor FileStream para indicar los siguientes parámetros:
- - El nombre del archivo que vamos a abrir.
- - El modo en el vamos a abrir el fichero (FileMode).
- - El modo en el que accedemos al fichero (FileAccess):
    - **Read**: acceso para leer el archivo.
    - **Write**: acceso de escritura al archivo.
    - **ReadWrite**: acceso de lectura y escritura al archivo.

Directiva using

Otra forma de hacer lo mismo que en el ejemplo de la página siguiente sin necesidad de fs.close ni fichero.close

- Es simplemente una forma de indicar que un objeto se va a utilizar únicamente en un bloque de código y que una vez finalizado éste debe ser "destruido".

- Esto se puede aplicar al trabajo con el flujo de los ficheros

Si dejamos el flujo abierto nos va a dar un error.

```
0 referencias
static void Main(string[] args)
{
    string fileName = @"mytest.txt";

    {
        Console.Write("\n\n Create a file named mytest.txt in the disk :\n");
        Console.Write("-------------------------------------------------\n");
        // Create the file.
        using (FileStream fileStr = File.Create(fileName))
        {
            Console.WriteLine(" A file created with name mytest.txt\n\n");
        }
    }


}
```

ILERNA Online

FileStream se debe utilizar siempre que trabajemos con ficheros secuenciales.

Ejemplo

variable de tipo fichero

es el nombre que le asignamos y que significa que va a ser de tipo texto. Es una ruta relativa, podemos poner c:\ilerna\projects

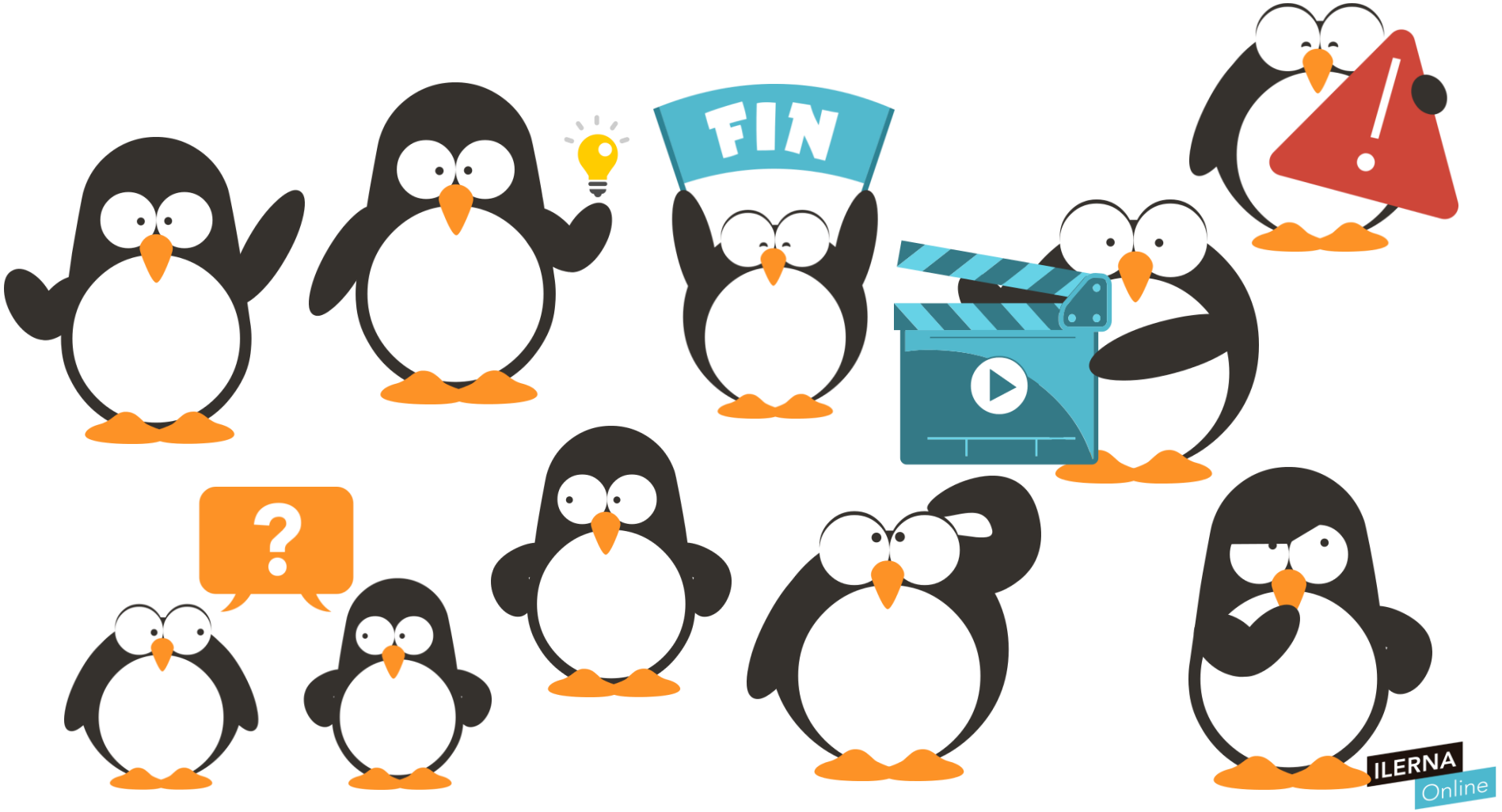
```
FileStream fichero = new FileStream("ejemplo.txt",FileMode.Open,
FileAccess.Read);
```
abrimos en modo lectura.

```
StreamReader fs = new StreamReader(fichero);
```
Tipo de pasarela que nosotros queremos, en este caso de lectura. Creamos la variable fs de tipo streamreader.

```
string line;
while((line = fs.ReadLine()) != null) {
 System.Console.WriteLine(line);
}
fs.close();
fichero.Close();
```

- En este ejemplo se lee el contenido de un archivo de texto línea a línea en una cadena mediante el método ReadLine de la clase StreamReader.
- Cada línea de texto se almacena en la cadena line y se muestra en la pantalla.

ILERNA Online


FIN
?
ILERNA
Online