

# CURSO DE PROGRAMACIÓN.NET

M.374.001.003



## ACCESO A FICHEROS CONTROL DE ERRORES



## Ejercicios

---

1. A partir de un archivo de texto que contenga un número en cada línea. Lee todos los números y calcula la suma. Muestra los números separados por '+' y el resultado final.
2. Crea un programa que vaya leyendo lo que el usuario escribe en consola y lo escriba en un fichero (línea a línea). Terminará cuando el usuario escriba la palabra 'FIN'.
3. Lee los datos de un fichero que contenga en cada línea el nombre de un alumno y su nota, separadas por punto y coma (Ejemplo → Pepito;6.75). Al final del todo informa de la media, quien tiene la nota más alta y quien la más baja.
4. Crea un programa que muestre el siguiente menú:
  - 1) Mostrar productos
  - 2) Añadir producto
  - 0) Salir

Trabajaremos con un fichero que contendrá la información de varios productos. Un producto en cada línea con los datos nombre y precio separados por punto y coma. La opción 1 mostrará los productos del fichero (formatea la salida para que los precios salgan alineados con 2 decimales). La opción 2 te pedirá el nombre de un producto y el precio y lo insertará al final del archivo.

Debes mostrar el menú hasta que el usuario seleccione salir. Cada una de las opciones impleméntalas en funciones separadas que llamarás desde el Main.

5. A partir de un archivo que contiene una palabra en cada línea. Carga las palabras en un array y selecciona una al azar. Pide al usuario que la adivine. Tiene 3 intentos.
6. Haz lo mismo que el ejercicio anterior pero en lugar de pedir que adivine la palabra directamente, muestra una cantidad de asteriscos de igual longitud que la palabra elegida (pista: puedes concatenar tantos asteriscos en una cadena nueva como longitud tenga la palabra). A continuación pide al usuario una letra repetidamente:
  - a. Si la letra está en la palabra, sustituye los asteriscos por dicha letra en la posición correspondiente. Cuando las palabras sean iguales habrá ganado.
  - b. Si no está, informa al usuario que ha fallado y muestra cuantos intentos le quedan. En total tiene 7.

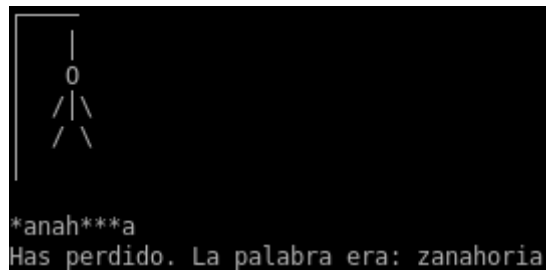
```
Palabra: ****
Dime letra: a
Palabra: *a*a
```

```
Dime letra: s
Palabra: *asa
Dime letra: c
Palabra: casa
Enhorabuena!, has acertado
```

7. **Reto:** A partir del programa anterior vamos a hacer el juego del ahorcado.

Primero limpia la pantalla y muestra la horca (usa guiones y barras). Para limpiar la pantalla usa **Console.Clear()**. Después muestra los asteriscos y pide al usuario una letra. Para recolocar el cursor en la consola puedes usar la función **Console.SetCursorPosition(columna, fila)**.

- Si acierta, coloca el cursor en la línea de la palabra y reescríbela con las letras sustituidas que toque. Si ha acertado todas las letras felicítale y sal.
- Si falla, dependiendo el número de fallo, tendrás que colocar el cursor en una posición de la horca y dibujar un carácter del muñeco (usa la estructura switch para ello). Después sitúa el cursor debajo de la palabra a adivinar. Si ha agotado todos los intentos le tienes que decir que ha perdido y salir.



```

  |
  0
 / \
/   \
/     \

*anah***a
Has perdido. La palabra era: zanahoria
```

8. Haz un programa que pregunte al usuario por el nombre de un fichero y muestra su contenido. Observa que cuando el fichero no existe lanza una excepción. Captúrala e informa de que el fichero no existe al usuario.

1. Crea un fichero de texto llamado "textFile1.txt". Escribe una línea con "Hola" en él. Escribe una línea con "Adiós" en él. Prueba el fichero resultante. Por ejemplo pulsando doble clic sobre él
2. Crea un fichero de texto llamado "textFile2.txt". Escribe en él 100 líneas "Línea 1", "Línea 2"... Prueba el fichero resultante.
3. Crea un fichero de texto llamado "primos.txt". Escribe los primeros 50 números primos, todos deben estar en la misma línea separados por espacios. Prueba el fichero resultante
4. El fichero llamado "textFile2.txt" contiene 100 líneas "Línea 1", "Línea 2"... Añade 100 línea más siguiendo la secuencia.
5. Pide un nombre de un fichero al usuario y saca por pantalla las líneas que comiencen por A o a.  
Número de línea: contenido de línea.
6. Crea un programa que pida el nombre de un fichero al usuario, compruebe si existe y muestre a continuación cuantas líneas tiene dicho fichero. (2 métodos, usando ReadLine y usando ReadAllLines)
7. Crea un programa que vaya pidiendo frases al usuario hasta que este pulse enter y las vaya guardando en un fichero. Debe comprobar si existe primero y si es así añadir las frases al final de las ya existentes.
8. Crea un programa que pida al usuario un nombre de fichero y muestre el número de palabras que contiene.
9. Lee un fichero de texto y almacénalo en un array (ReadAllLines). Pídele al usuario si desea verlo al revés o no, y muéstralo por pantalla como lo haya solicitado el usuario.
10. Crea un programa que se comporte como el comando "more" de Unix: Debe mostrar el contenido de un fichero y pedir al usuario que pulse Enter cada vez que la pantalla esté llena.  
Como una aproximación sencilla, mostraremos las línea truncadas en 79 caracteres, y pararemos después de 24 líneas.

## EJERCICIOS DE FICHEROS BINARIOS

1. Cree un programa en C# que lea las etiquetas de la especificación ID3 v1 desde un archivo de música MP3. También debería comprobar si la etiqueta ID corresponde con los caracteres TAG de la versión 1.

Las especificaciones ID3 se aplican a cualquier archivo o contenedor audiovisual. Sin embargo, generalmente se aplica principalmente contenedores de audio. Hay tres versiones de la especificación que son compatibles. Por ejemplo, un archivo puede contener etiquetas simultáneamente versión 1.1 y versión 2.0. En este caso, el reproductor multimedia debe decidir cuáles son relevantes.

### ID3 version 1

Esta primera especificación es muy simple. Consiste en adjuntar un bloque de tamaño fijo de 128 bytes al final del fichero en cuestión. Este bloque contiene las siguientes etiquetas:

- Una cabecera de identificación con los caracteres "TAG".
- Título: 30 caracteres.
- Artista: 30 caracteres.
- Álbum: 30 caracteres.
- Año: 4 caracteres.
- Un comentario: 30 caracteres.
- Género (musical): un carácter.

Todas las etiquetas usan caracteres ASCII, excepto el género, que es un número entero almacenado en un único byte. El género musical asociado a cada byte está predefinido en el estándar e incluye definiciones de 80 géneros, numerados del 0 al 79.

Determinados programas de reproducción han ampliado por su cuenta los géneros definidos (a partir del número 80).

### Salida

```
TAG
Impact Moderato
Nauj Llopier
YouTube Audio Library
1993
```

2. Cree un programa en C# para leer las dimensiones de una imagen con formato Windows bitmap. Primero debería comprobar que se trata de una imagen .bmp válida, revisando los datos de cabecera 'BM'. Si se trata de una imagen .bmp válida entonces obtenga sus dimensiones (ancho x alto) y muéstrelas en pantalla.

Es un formato propio del sistema operativo Windows. Puede guardar imágenes hasta 24 bits (16,7 millones de colores).

El encabezado de una imagen BMP es el siguiente:

Descripción	Bytes
-------------	-------

Tipo (BM)	0-1
Tamaño	2-5
Reservado	6-9
Inicio de los datos de la imagen	10-13
Tamaño del bitmap	14-17
Ancho (píxeles)	18-21
Alto (píxeles)	22-25
Número de planos	26-27
Tamaño de cada punto	28-29
Compresión	30-33
Tamaño de imagen	34-37
Resolución horizontal	38-41
Resolución vertical	42-45
Tamaño de la tabla de color	46-49
Contador de colores	50-53

## Salida

48x48