

Ejercicios Concurrencia

[Descargar estos ejercicios](#)


Índice

1. [Ejercicio 1](#)
2. [Ejercicio 2](#)
3. [Ejercicio 3](#)
4. ☒ [Ejercicio 4](#)
5. [Ejercicio 5](#)

Para realizar los siguientes ejercicios deberás usar el siguiente método...

```
[MethodImpl(MethodImplOptions.Synchronized)]
static void SituaYDibuja(
    int col, int fila,
    string texto,
    ConsoleColor colorFondo, ConsoleColor colorLetra)
{
    Console.CursorVisible = false;
    Console.SetCursorPosition(col, fila);
    Console.BackgroundColor = colorFondo;
    Console.ForegroundColor = colorLetra;
    Console.Write(texto);
    Console.ResetColor();
}
```

Al tener atributo **Synchronized**, evita que si tenemos **varias tareas** o hilos en nuestro programa ejecutándose al mismo tiempo, estas puedan ejecutar el código de este método estático al mismo tiempo. De esta manera, evitaremos el problema de que mientras un hilo esta haciendo un Write otro esté cambiando el cursor de posición al mismo tiempo.

 **Aviso:** Necesitaremos incluir using **System.Runtime.CompilerServices**

Ejercicio 1

Vamos a crear una aplicación de consola que su programa principal lance un temporizador que cada segundo muestre en la parte superior derecha de la pantalla la hora actual del sistema en formato **HH:MM:SS**.


Tras lanzar el temporizador el programa principal se quedará esperando una pulsación de teclado con **ReadKey()** y si pulsamos **ESC** el temporizador parará y saldremos del programa. **Pero si pulsamos cualquier otra tecla el temporizador parará o se reiniciará según su estado anterior.**

Ejercicio 2

Vamos a ampliar el ejercicio anterior añadiendo música a nuestra aplicación, para ello, después de lanzar el Timer, intanciaremos un objeto de la clase **System.Media.SoundPlayer**, al cual pasaremos un archivo de audio con extensión **.wav** y codec **PCM** que habrás conseguido previamente.

Dicha clase tiene un método **PlayLooping()** que empezará a reproducir el archivo en un hilo aparte. Al igual que antes, al pulsar cualquier tecla, tanto el timer como la música pararán y,

volverán a reanudarse, si se vuelve a pulsar una tecla. Como en el ejercicio anterior, el programa parará si pulsamos ESC.

 **Aviso:** Ten en cuenta que tanto **Timer** como **SounPlayer** son disponibles y por tanto deberás instanciarlos dentro de una clausula **using**.

 **Importante:** Si no se reconoce **System.Media** podría ser necesario añadir el paquete **System.Windows.Extensions** descargandolo de Nuget, como ya vimos en temas anteriores

Ejercicio 3


Crea una aplicación que lea un fichero que podrás encontrar junto a este enunciado y que se llama **quijote_reducido.txt**.

En el programa principal (Main) deberás crear un **Task** (usando una función o expresión Lambda) que lea carácter a carácter el fichero y que añada en un diccionario la cantidad de cada una de las vocales que contiene usando como clave el carácter con la vocal pasado a mayúscula.

La tarea no recibirá nada y retornará dicho diccionario con las apariciones de cada vocal.

Una vez lse lance la tarea, el programa principal se quedará en bucle cuya condición de salida sea la comprobación de la tarea y mostrará cada segundo la fecha y la hora en la esquina superior izquierda.

Al finalizar la cuenta y por tanto salir del bucle, se mostrará el diccionario ordenado por clave. *(Usa la función **OrderBy**).*

 **Aviso:** Si la cuenta la realiza muy rápido puedes meter una pausa de 2 o 3 milisegundos entre cada vocal leída.

```
15/05/2018 20:12:11
Finalizada cuenta de letras...
La vocal A aparece 351 veces.
La vocal E aparece 444 veces.
La vocal I aparece 249 veces.
La vocal O aparece 306 veces.
La vocal U aparece 122 veces.
Presione una tecla para continuar . . .
```


Ejercicio 4

Partiendo del ejercicio anterior vamos a modificar su diseño y funcionalidad para que cumpla las siguientes condiciones:

- Vamos a añadir un temporizador con la hora como el del primer ejercicio que me muestre la hora mientras se realiza la contabilización de la hora. Para ello crea una función **static Timer CreaTemporizadorReloj()** que muestre la hora y inicialice la temporalización para que se vuelva a mostrar cada segundo. El temporizador será devuelto para pararlo y destruirlo de forma adecuada.
- Vamos a poder cancelar el proceso de contabilización de vocales y para ello definiremos la siguiente clase de la que pasaremos un objeto instanciado de la misma a la tarea para notificarle que queremos cancelarla.

```
```csharp
class FlagTaskCancelled {
public bool IsCancelled { get; set; }
}
```
```

- La creación de la tarea la vamos a sacar del programa principal y la vamos a encapsular en un método llamado `static Task LeeVocalesAsync(FlagTaskCancelled finalizaLectura)` el cual devolverá una tarea que **ya no retornará nada** ya que el diccionario se irá mostrando en la misma tarea conforme se contabilizan la variables. Además la tare, ahora sí **recibirá un dato** que será el objeto **FlagTaskCancelled finalizaLectura** que le permitirá comprobar una solicitud de cancelación del proceso de contabilización.
- El programa principal se suscribirá al evento **CancelKeyPress** de la consola en cuyo manejador pondremos a true el flag que hemos pasado a la tarea y detendremos la propagación del evento de cancelación.

 **Aviso:** Además, el programa principal ya no mostrará ninguna hora porque esta ya es mostrada por el nuevo temporizador y simplemente se limitará a realizar un **Wait** de la tarea de contabilización, tras suscribirse al evento de cancelación.

Pulsa Control + C para interrumpir el proceso.

12:06:31 58%

Vocales contabilizadas hasta el momento...

La vocal A aparece 187 veces.
 La vocal E aparece 267 veces.
 La vocal I aparece 142 veces.
 La vocal O aparece 188 veces.
 La vocal U aparece 69 veces.

Ejercicio 5

Vamos a realizar un programa de consola que se syndique a contenidos RSS de la sección internacional de algunos periódicos españoles filtrando por una palabra clave.

1. Prediremos la palabra para filtrar titulares. Ej: Nombre presidente EE.UU
2. Definiremos un diccionario donde la clave sea el nombre del diario y el valor la URL con el XML a syndicar. Por ejemplo:

El Mundo → <https://e00-elmundo.uecdn.es/elmundo/rss/internacional.xml>

ABC → http://www.abc.es/rss/feeds/abc_Internacional.xml

El Diario → <https://www.eldiario.es/rss/section/4000/>

Nota: Estos enlaces pueden estar obsoletos, pruébalos por si hay que actualizarlos.

3. Para realizar las tareas asíncronas deberemos usar las cláusulas **async y await**.
4. Para filtrar los titulares extraeremos mediante una ER el contenido entre las etiquetas `<title> ... </title>` en algunas sindicaciones podremos encontrar la `<title><![CDATA[...]]></title>` que podremos tener en cuenta en la propia ER o eliminar de la cadena resultante mediante un **mapeo** después del filtrado.
5. El programa principal se quedará en un bucle que indique que se está haciendo la búsqueda.