9. Algunas bibliotecas adicionales de uso frecuente

9.1. Fecha y hora. Temporización

Desde C#, tenemos la posibilidad de manejar **fechas y horas** con facilidad. Para ello, tenemos el tipo de datos DateTime. Por ejemplo, podemos hallar la fecha (y hora) actual con:

```
DateTime fecha = DateTime.Now;
```

O crear un objeto DateTime a partir de una cierta fecha usando el constructor:

```
DateTime fecha = new DateTime(1990, 9, 18);
```

Dentro de ese tipo de datos DateTime, tenemos herramientas para saber el día (Day), el mes (Month) o el año (Year) de una fecha, entre otros, que veremos con detalle un poco más adelante. También podemos calcular otras fechas sumando a la actual una cierta cantidad de segundos (AddSeconds), días (AddDays), etc. Un ejemplo básico de su uso sería:

```
// Ejemplo_09_01a.cs
// Ejemplo básico de manejo de fechas
// Introducción a C#, por Nacho Cabanes

using System;

class Ejemplo_09_01a
{
    static void Main()
    {
        DateTime fecha = DateTime.Now;
        Console.WriteLine("Hoy es {0} del mes {1} de {2}",
            fecha.Day, fecha.Month, fecha.Year);
        DateTime manyana = fecha.AddDays(1);
        Console.WriteLine("Mañana será {0}",
            manyana.Day);
    }
}
```

Aunque también se puede escribir

```
Console.WriteLine("Fecha actual: "+ DateTime.Now);
```

Y en ese caso, aparecerán día, mes, año, hora, minutos y segundos en el formato regional que se haya escogido en el sistema operativo, que en España es de esperar que sea algo como 16/04/2018 11:01:02

Algunas de las propiedades más útiles del tipo de datos DateTime son:

- Now (fecha y hora actual de este equipo)
- Today (fecha actual, sin hora)
- Day (día del mes)
- Month (número de mes)
- Year (año)
- Hour (hora)
- Minute (minutos)
- Second (segundos)
- Millisecond (milisegundos)
- DayOfYear (día del año)
- DayOfWeek (día de la semana: su nombre en inglés, que se puede convertir en un número del 0 -domingo- al 6 -sábado- si se fuerza su tipo a entero, anteponiéndole "(int)").

Si sólo se desea mostrar fechas en formatos concretos, no hace falta acceder a cada una de esas propiedades, sino que se puede emplear directamente ToString, con varios formatos predefinidos, como

```
using System;

class FormatosDeFecha
{
    static void Main()
    {
        DateTime unaFecha = new DateTime(1990, 12, 31);
        Console.WriteLine(unaFecha);

        Console.WriteLine("d= " + unaFecha.ToString("d"));
        Console.WriteLine("D= " + unaFecha.ToString("D"));
        Console.WriteLine("g= " + unaFecha.ToString("g"));
        Console.WriteLine("t= " + unaFecha.ToString("t"));
    }
}
```

Que mostraría como resultado:

```
31/12/1990 0:00:00
d= 31/12/1990
D= lunes, 31 de diciembre de 1990
g= 31/12/1990 0:00
t= 0:00
```

Puedes encontrar más detalles sobre los formatos existentes en la referencia oficial de Microsoft, MSDN:

https://docs.microsoft.com/es-es/dotnet/api/system.datetime.tostring?view=netframework-4.8

Para calcular nuevas fechas a partir de una dada, podemos usar métodos que generan un nuevo DateTime, como:

- AddDays (que aparece en el ejemplo anterior), AddHours, AddMilliseconds, AddMinutes, AddMonths, AddSeconds, AddHours
- También un Add más genérico (para sumar una fecha a otra) y un Subtract también genérico (para restar una fecha de otra).

Cuando restamos dos fechas, obtenemos un dato de tipo "intervalo de tiempo" (TimeSpan), del que podemos saber detalles como la cantidad de días (sin decimales, Days, o con decimales, TotalDays), como se ve en este ejemplo:

```
// Ejemplo_09_01b.cs
// Diferencia entre dos fechas
// Introducción a C#, por Nacho Cabanes

using System;

class Ejemplo_09_01b
{
    static void Main()
    {
        DateTime fechaActual = DateTime.Now;
        DateTime fechaNacimiento = new DateTime(1990, 9, 18);

        TimeSpan diferencia =
            fechaActual.Subtract(fechaNacimiento) ;

        Console.WriteLine("Han pasado {0} días",
            diferencia.Days);

        Console.WriteLine("Si lo quieres con decimales: {0} días",
            diferencia.TotalDays);

        Console.WriteLine("Y si quieres más detalles: {0}",
            diferencia);
    }
}
```

El resultado de este programa sería algo como

```
Han pasado 8886 días
Si lo quieres con decimales: 8886,41919806277 días
Y si quieres más detalles: 8886.10:03:38.7126236
```

También podemos hacer una **pausa** en la ejecución: si necesitamos que nuestro programa se detenga una cierta cantidad de tiempo, no hace falta que usemos un "while" que compruebe la hora continuamente, sino que podemos "bloquear" (Sleep) el subproceso (o hilo, "Thread") que representa nuestro programa una cierta cantidad de milésimas de segundo con: Thread.Sleep(5000);

Este método pertenece a System.Threading, que deberíamos incluir en nuestro apartado de"using", o bien emplear la llamada completa:

```
// Ejemplo_09_01c.cs
// Pausas
// Introducción a C#, por Nacho Cabanes

using System;
using System.Threading;

class Ejemplo_09_01c
{
    static void Main()
    {
        DateTime fecha = DateTime.Now;
        Console.WriteLine("Son las {0}:{1}:{2}",
            fecha.Hour, fecha.Minute, fecha.Second);
        Console.WriteLine("Vamos a esperar 3 segundos...");
        Thread.Sleep(3000);
        fecha = DateTime.Now;
        Console.WriteLine("Ahora son las {0}:{1}:{2}",
            fecha.Hour, fecha.Minute, fecha.Second);
    }
}
```

Ejercicios propuestos:

(9.1.1) Crea una versión mejorada del ejemplo 10_01a, que muestre el nombre del mes (usa un array para almacenar los nombres y accede a la posición correspondiente), la hora, los minutos, los segundos y las décimas de segundo (no las milésimas). Si los minutos o los segundos son inferiores a 10, deberán mostrarse con un cero inicial, de modo que siempre aparezcan con dos cifras.

(9.1.2) Crea un reloj que se muestre en pantalla, y que se actualice cada segundo (usando "Sleep"). En esta primera aproximación, el reloj se escribirá con "WriteLine", de modo que aparecerá en la primera línea de pantalla, luego en la segunda, luego en la tercera y así sucesivamente (en el próximo apartado veremos cómo hacer que se mantenga fijo en unas ciertas coordenadas de la pantalla).

(9.1.3) Crea un programa que pida al usuario su fecha de nacimiento, y diga de qué día de la semana se trataba, usando ".DayOfWeek".
(9.1.4) Crea un programa que muestre el calendario del mes actual (pista: primero deberás calcular qué día de la semana es el día 1 de este mes). Deberá ser algo como:

```
    Mayo 2020

lu ma mi ju vi sá do
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

(9.1.5) Crea un programa que muestre el calendario correspondiente al mes y el año que se indiquen como parámetros en línea de comandos.
(9.1.6) Crea un programa que vuelque a un fichero de texto el calendario del año que se indique como parámetro en línea de comandos. Deben mostrarse tres meses en anchura: el primer bloque contendrá enero febrero y marzo, el segundo tendrá abril, mayo y junio y así sucesivamente.
(9.1.7) Crea un programa que pregunte al usuario el día y mes en que nació, y le diga cuántos días faltan hasta su próximo cumpleaños.