INAP | CAMPUS VIRTUAL

Programación II

Ministerio de Modernización
Presidencia de la Nación

Módulo II

Programación II

Módulo II: Material extra

Módulo II:

- **Otras posibilidades de la “consola”**

Otras posibilidades de la "consola"

En "Console" hay mucho más que ReadLine y WriteLine, aunque no todas las posibilidades están incluidas en las primeras versiones de la "plataforma punto net". Vamos a ver algunas de las funcionalidades adicionales que nos pueden resultar más útiles:

- Clear: borra la pantalla.
- ForegroundColor: cambia el color de primer plano (para indicar los colores, hay definidas constantes como "ConsoleColor.Black", que se detallan al final de este apartado).
- BackgroundColor: cambia el color de fondo (para el texto que se escriba a partir de entonces; si se quiere borrar la pantalla con un cierto color, se deberá primero cambiar el color de fondo y después usar "Clear").
- SetCursorPosition(x, y): cambia la posición del cursor ("x" se empieza a contar desde el margen izquierdo, e "y" desde la parte superior de la pantalla).
- Readkey(interceptar): lee una tecla desde teclado. El parámetro "interceptar" es un "bool", y es opcional. Indica si se debe capturar la tecla sin permitir que se vea en pantalla ("true" para que no se vea, "false" para que se pueda ver). Si no se indica este parámetro, la tecla se muestra en pantalla.
- KeyAvailable: indica si hay alguna tecla disponible para ser leída (es un "bool")
- Title: el título que se va a mostrar en la consola (es un "string")

```
using System;

public class Ejemplo_12_02a
{
    public static void Main()
    {
        int posX, posY;

        Console.Title = "Ejemplo de consola";
        Console.BackgroundColor = ConsoleColor.Green;
        Console.ForegroundColor = ConsoleColor.Black;
        Console.Clear();

        posY = 10;  // En la fila 10
        Random r = new Random(DateTime.Now.Millisecond);
        posX = r.Next(20, 40); // Columna al azar entre 20 y 40
        Console.SetCursorPosition(posX, posY);
        Console.WriteLine("Bienvenido");


        Console.ForegroundColor = ConsoleColor.Blue;
        Console.SetCursorPosition(10, 15);
        Console.Write("Pulsa 1 o 2: ");
        ConsoleKeyInfo tecla;
        do
        {
            tecla = Console.ReadKey(false);
        }
        while ((tecla.KeyChar != '1') && (tecla.KeyChar != '2'));

        int maxY = Console.WindowHeight;
        int maxX = Console.WindowWidth;
        Console.SetCursorPosition(maxX-50, maxY-1);
        Console.ForegroundColor = ConsoleColor.Red;
        Console.Write("Pulsa una tecla para terminar... ");
        Console.ReadKey(true);
    }
}
```

(Nota: no todas las posibilidades que se aplican en este ejemplo están disponibles en la plataforma .Net 1.x, sino a partir de la versión 2).

Para comprobar el valor de una tecla, como se ve en el ejemplo anterior, tenemos que usar una variable de tipo "ConsoleKeyInfo" (información de tecla de consola). Un ConsoleKeyInfo tiene campos como:

- KeyChar, que representa el carácter que se escribiría al pulsar esa tecla. Por ejemplo, podríamos hacer if (tecla.KeyChar == '1') ...
- Key, que se refiere a la tecla (porque hay teclas que no tienen un carácter visualizable, como F1 o las teclas de cursor). Por ejemplo, para comprobar la tecla ESC podríamos hacer if (tecla.Key == ConsoleKey.Escape) ... . Algunos de los códigos de tecla disponibles son:
    - Teclas de edición y control como, como: Backspace (Tecla RETROCESO), Tab (Tecla TAB), Clear (Tecla BORRAR), Enter (Tecla ENTRAR), Pause (Tecla PAUSA), Escape (Tecla ESC (ESCAPE)), Spacebar (Tecla BARRA ESPACIADORA), PrintScreen (Tecla IMPR PANT), Insert (Tecla INS (INSERT)), Delete (Tecla SUPR (SUPRIMIR))

    - Teclas alfabéticas, como: A (Tecla A), B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

- Teclas numéricas, como: D0 (Tecla 0), D1, D2, D3, D4, D5, D6, D7, D8, D9

- Teclado numérico adicional: NumPad0 (Tecla 0 del teclado numérico), NumPad1, NumPad2, NumPad3, NumPad4, NumPad5, NumPad6, NumPad7, NumPad8, NumPad9, Multiply (Tecla Multiplicación), Add (Tecla Suma), Separator (Tecla Separador), Subtract (Tecla Resta), Decimal (Tecla Decimal), Divide (Tecla División)
- Sleep (Tecla Espera del equipo)
- Teclas de función: F1, F2 y sucesivas (hasta F24)
- Teclas especiales de Windows: LeftWindows (Tecla izquierda con el logotipo de Windows), RightWindows (Tecla derecha con el logotipo de Windows)
- Incluso teclas multimedia, si el teclado las incorpora, como: VolumeMute (Tecla Silenciar el volumen, enMicrosoft Natural Keyboard, bajo Windows 2000 o posterior), VolumeDown (Bajar el volumen, ídem), VolumeUp (Subir el volumen), MediaNext (Tecla Siguiente pista de multimedia), etc.
- Modifiers, que permite comprobar si se han pulsado simultáneamente teclas modificadoras: Alt, Shift o Control. Un ejemplo de su uso sería:

```
if ((tecla.Modifiers & ConsoleModifiers.Alt) != 0)
    Console.Write("Has pulsado Alt");
```

Console.Readkey hace que el programa se quede parado hasta que se pulse una tecla. Si queremos hacer algo mientras que el usuario no pulse ninguna tecla, podemos emplear Console.KeyAvailable para comprobar si ya se ha pulsado alguna tecla que haya que analizar, como en este ejemplo, que permite mover un símbolo a izquierda y derecha, pero muestra una animación simple si no pulsamos ninguna tecla:

```
using System;
using System.Threading;

public class Ejemplo_12_02b
{
    public static void Main()
    {
        int posX=40, posY=10;
        string simbolos = "^>v<";
        byte simboloActual = 0;
        bool terminado = false;

        do
        {
            Console.Clear();
            Console.SetCursorPosition(posX, posY);
            Console.Write( simbolos[ simboloActual ]);
            Thread.Sleep(500);
            if (Console.KeyAvailable)
            {
                ConsoleKeyInfo tecla = Console.ReadKey(true);
                if (tecla.Key == ConsoleKey.RightArrow) posX++;
                if (tecla.Key == ConsoleKey.LeftArrow) posX--;
                if (tecla.Key == ConsoleKey.Escape) terminado = true;
            }
            simboloActual++;
            if (simboloActual > 3) simboloActual = 0;
        }
        while ( ! terminado );
    }
}
```

Al igual que en este ejemplo, será recomendable hacer una pequeña pausa entre una comprobación de teclas y la siguiente, con Thread.Sleep, tanto para que la animación no sea demasiado rápida como para no hacer un consumo muy alto de procesador para tareas poco importantes.

Los **colores** que tenemos disponibles (y que se deben escribir precedidos con "ConsoleColor") son: Black (negro), DarkBlue (azul marino), DarkGreen (verde oscuro) DarkCyan (verde azulado oscuro), DarkRed (rojo oscuro), DarkMagenta (fucsia oscuro o púrpura), DarkYellow (amarillo oscuro u ocre), Gray (gris), DarkGray (gris oscuro), Blue (azul), Green (verde), Cyan (aguamarina o verde azulado claro), Red (rojo), Magenta (fucsia), Yellow (amarillo), White (blanco).