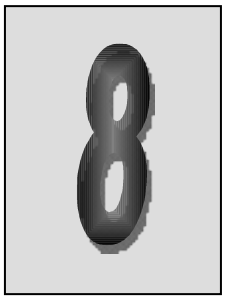


Aplicaciones de Consola.

1.- Introducción.

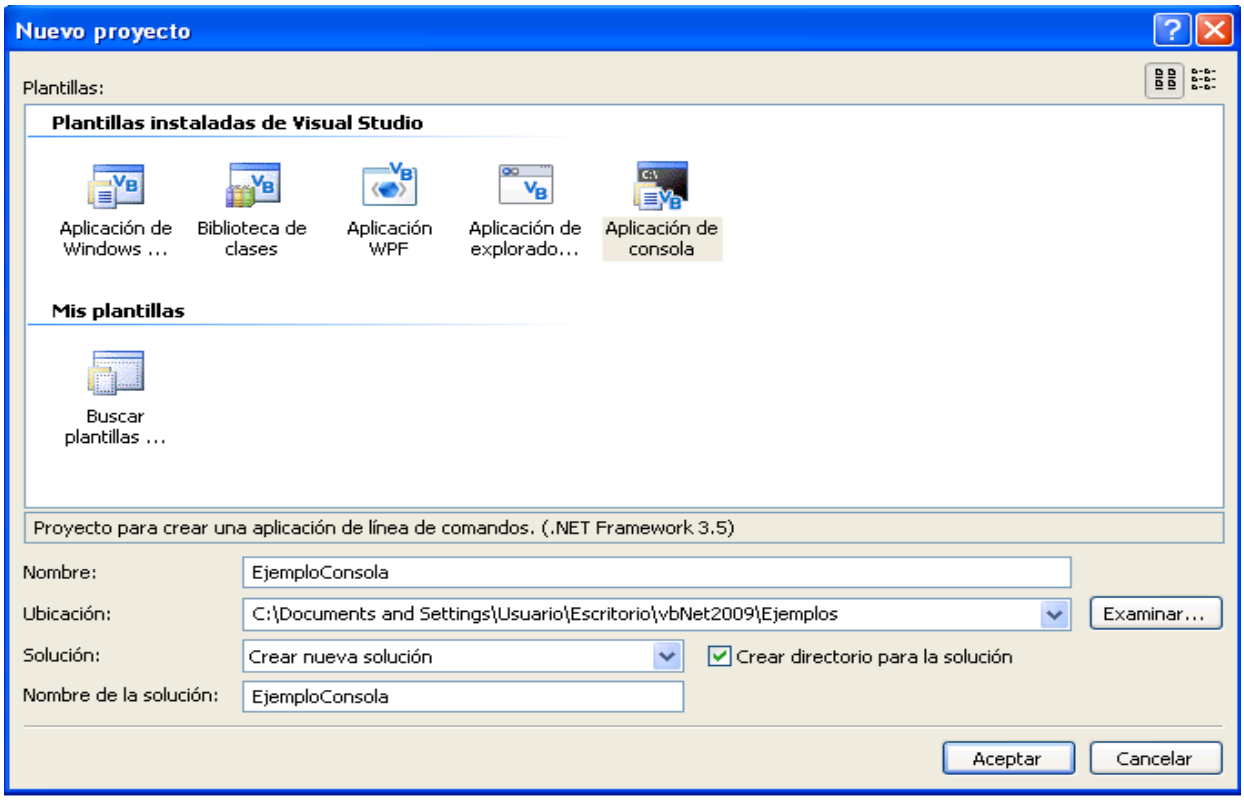
Una aplicación de consola es aquella que se ejecuta dentro de una ventana de línea de comandos. Este tipo de ventana recibe diferentes denominaciones: Símbolo del sistema, Sesión MS-DOS, Ventana de línea de comandos, etc., por lo que a lo largo de este capítulo se hará referencia a ella de forma genérica como consola.

Las aplicaciones de consola son muy útiles cuando se necesitan realizar pruebas que no impliquen el uso del modo gráfico del sistema operativo: formularios, controles, imágenes, etc., ya que consumen menos recursos y su ejecución es más veloz.

En este caso particular, debido a que los próximos temas versarán sobre aspectos del lenguaje, y en ellos no se necesitará obligatoriamente el uso de formularios, se utilizará aplicaciones de consola para los ejemplos.

2.- Creación de un proyecto de tipo aplicación de consola.

Para crear una aplicación de consola básica, después de iniciar el IDE de VS.NET, y seleccionar el menú para crear un nuevo proyecto, elegir **Aplicación de consola** en el panel derecho de la ventana **Nuevo proyecto**. El resto de opciones de esta ventana se configuran igual que para una aplicación con formularios Windows.



Después de pulsar Aceptar se creará el proyecto que contendrá un fichero de código con el nombre MODULE1.VB, en cuyo interior se encontrará un módulo de código conteniendo un procedimiento Main( ) vacío, por el que comenzará la ejecución del programa.

```
Module Module1

    Sub Main()

    End Sub

End Module
```

3.- La clase Console.

Esta clase se encuentra dentro del espacio de nombres System, y proporciona a través de sus métodos, acceso a la consola para mostrar u obtener información del usuario.

Debido a que los miembros de esta clase se encuentran compartidos (shared), no es necesario crear una instancia previa de la misma en una variable, pudiendo ejecutar directamente sus métodos sobre el objeto Console. Todo ello se explicará en los siguientes apartados.

4.- Escritura de información.

Para mostrar texto utilizar el método **WriteLine( )** del objeto Console. Este método escribe en la línea actual de la consola el valor que le paserá como parámetro, añadiendo automáticamente las marcas de retorno de carro y nueva línea, por lo que la siguiente escritura se realizará en una nueva línea.

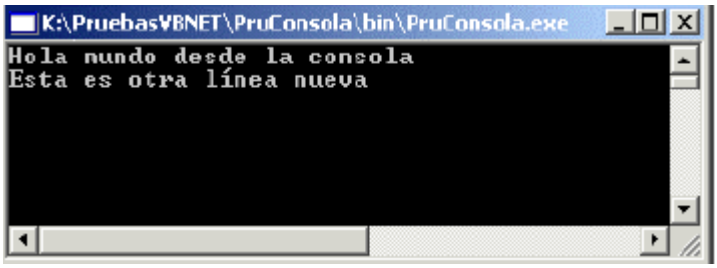
```
Sub Main()
    Console.WriteLine("Hola mundo desde la consola")
    Console.WriteLine("Esta es otra línea nueva")
End Sub
```

El código fuente anterior tiene no obstante un inconveniente: cuando el lector lo ejecute observará que se muestra la consola con las líneas de texto, pero inmediatamente vuelve a cerrarse, no dejando apenas tiempo para ver su contenido. Esto es debido a que no utiliza ninguna instrucción que establezca una parada en la ejecución, que permita observar el resultado de lo que se ha escrito en la consola.

Para remediar este problema, utilizar el método **ReadLine( ),** que realiza una lectura de los caracteres que se vayan introduciendo en la línea actual de la consola, e impedirá continuar la ejecución hasta que no se pulse [INTRO].

```
Sub Main()
    Console.WriteLine("Hola mundo desde la consola")
    Console.WriteLine("Esta es otra línea nueva")

    Console.ReadLine()
End Sub
```

Y el resultado es:



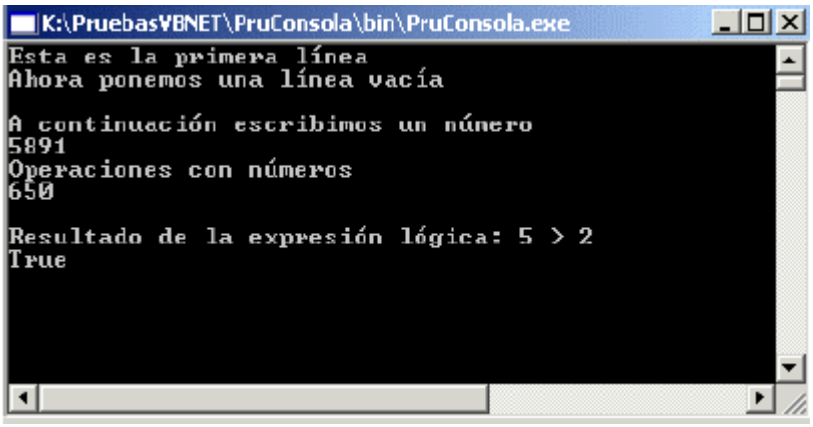
Los valores a mostrar con **WriteLine( )** pueden ser de distintos tipos de datos, pudiendo insertar también líneas en blanco.

```
Sub Main()
    ' ejemplos con WriteLine()
```

```
    ' escritura de cadenas de caracteres
    Console.WriteLine("Esta es la primera línea")
    Console.WriteLine("Ahora ponemos una línea vacía")

    Console.WriteLine()  ' línea vacía

    ' escritura de números
    Console.WriteLine("A continuación escribimos un número")
    Console.WriteLine(5891)

    Console.WriteLine("Operaciones con números")
    Console.WriteLine(500 + 150)

    Console.WriteLine()  ' otra línea vacia
```

```
    ' escritura de valores lógicos
    Console.WriteLine("Resultado de la expresión lógica: 5 > 2")
    Console.WriteLine(5 > 2)

    ' parada y espera respuesta del usuario
    Console.ReadLine()
End Sub
```

La siguiente imagen muestra la consola con el resultado de la ejecución del anterior fuente.



Write( ) es otro método permite también escribir valores en la consola. Su uso es igual que **WriteLine(),** aunque hay tener en cuenta que Write( ) no separa los valores a mostrar.

```
Sub Main()
    Console.Write("Hola")
    Console.Write("A")
    Console.Write("Todos")
    Console.Write(3456)
End Sub
```

La ejecución del anterior código mostraría en la consola los valores así: HolaATodos3456.

Para evitar que el texto en la consola salga junto, incluir espacios al comienzo y/o al final en las cadenas de caracteres que pasemos como parámetro a **Write( )**, o bien utilizar este método pasando una cadena vacía.

```
Sub Main()
    ' ejemplos con Write()

    Console.Write("Hola ")
    Console.Write("A")
    Console.Write(" Todos")
    Console.Write(" ")
    Console.Write(3456)

    Console.ReadLine()
End Sub
```

5.- Escritura de múltiples valores en la misma línea.

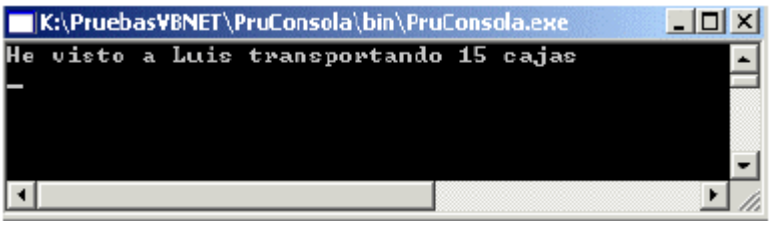
Al utilizar **WriteLine( )** o **Write( )** ocurrirá con frecuencia que en el texto a mostrar deba incluir valores que se encuentran en variables o expresiones, por lo que tendrás que realizar una combinación de la cadena de texto principal con los demás elementos para obtener la cadena final que mostrará al usuario. Esto se puede hacer empleando dos técnicas:

Concatenación.

Concatenando a la cadena principal las variables que contienen los valores a mostrar.

```
' concatenar múltiples valores

' declarar variables
Dim Nombre As String
Dim Numero As Integer

' asignar valor a las variables
Nombre = "Luis"
Numero = 15

Console.WriteLine("He visto a " & Nombre & " transportando " & Numero & " cajas")
Console.ReadLine()
```

Y el resultado es:



Parámetros sustituibles.

Pasando como primer parámetro la cadena a mostrar, y añadiendo tantos parámetros adicionales como valores debamos mostrar en la cadena principal. En la cadena principal indicar el lugar en donde visualizar los parámetros poniendo su número entre los símbolos de llave "{ }". El siguiente código fuente muestra la misma situación del ejemplo anterior pero utilizando esta técnica. El resultado final en la consola es el mismo que el del ejemplo anterior.

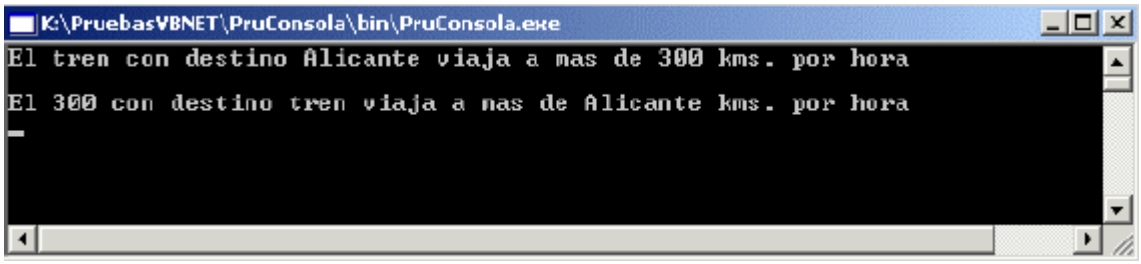
```
' combinación de múliples valores

' declarar variables
Dim Nombre As String
Dim Numero As Integer

' asignar valor a las variables
Nombre = "Luis"
Numero = 15

' el primer parámetro contiene la cadena a mostrar,
' el resto de parámetros son las variables que se visualizarán
' en alguna posición de la cadena del primer parámetro,
' el contenido de la variable Nombre se situará en el lugar
' indicado por {0}, la variable Numero en la posición de {1} y
' así sucesivamente
Console.WriteLine("He visto a {0} transportando {1} cajas", Nombre, Numero)
Console.ReadLine()
```

Como habrá comprobado el lector, los parámetros sustituibles comienzan a numerarse por cero, no estando obligados a mostrarlos en el mismo orden en el que los hemos situado en la llamada al método. El siguiente código fuente muestra dos ejemplos de sustitución de parámetros, uno de ellos con el mismo orden en el que se han situado en WriteLine( ), y otro con un orden distinto.

```
' declaración de variables
Dim Lugar As String
Dim Numero As Integer
Dim Vehiculo As String

' asignación de valor a variables
Vehiculo = "tren"
Lugar = "Alicante"
Numero = 300

' visualización de resultados en consola
Console.WriteLine("El {0} con destino {1} viaja a mas de {2} kms. por hora",
Vehiculo, Lugar, Numero)
```

```
Console.WriteLine()

Console.WriteLine("El {2} con destino {0} viaja a mas de {1} kms. por hora",
Vehiculo, Lugar, Numero)
Console.ReadLine()
```

Al ejecutar este fuente, el resultado será:



6.- Lectura de información

Para obtener el texto escrito por el usuario en la línea actual de la consola y hasta la pulsación de [INTRO] podemos utilizar el método **ReadLine( )** del objeto Console.

El siguiente código fuente muestra como volcar a una variable el contenido de la línea escrita por el usuario y posteriormente exponer su contenido, también a través de la consola.

```
' declaramos una variable para volcar el contenido
' de una línea de la consola
Dim LineaTexto As String

Console.WriteLine("Introducir un texto")
LineaTexto = Console.ReadLine()  ' el texto se pasa a la variable

' ahora mostramos lo que hemos escrito
Console.WriteLine()
Console.WriteLine("El usuario ha escrito la siguiente línea:")
Console.WriteLine(LineaTexto)

' aquí evitamos cerrar la consola,
' así podemos ver mejor el resultado
Console.ReadLine()
```

Read( ) es otro método del objeto Console que permite también la lectura del dispositivo de entrada de la consola, pero en este caso devuelve el código de una sola tecla pulsada por el usuario. Para ilustrar el uso de este método está el siguiente ejemplo, en el que después de pulsar varias teclas, nos introducimos en un bucle que va extrayendo cada uno de sus códigos, que volvemos a transformar en el carácter correspondiente a la tecla pulsada.

```
' ejemplo con Read()
Dim CodTecla As Integer
Dim NombreTecla As Char

Console.WriteLine("Pulsar varias teclas")
Console.WriteLine()

While True
    ' tomar los códigos de las teclas uno a uno
    CodTecla = Console.Read()

    ' si se ha pulsado intro, salir
    If CodTecla = 13 Then
        Exit While
    End If

    Console.WriteLine("Código de tecla pulsada: {0}", CodTecla)

    ' convertir el código al caracter de la tecla
    NombreTecla = Chr(CodTecla)

    Console.WriteLine("Tecla pulsada: {0}", NombreTecla)
End While

Console.ReadLine()
Console.WriteLine("Ejemplo terminado, pulse intro")
Console.ReadLine()
```

ReadKey( ) permite pulsar una tecla, este método se utiliza para el mensaje "Pulse cualquier tecla para continuar …·

```
Console.Write("Pulse cualquier tecla  para continuar ...")
Console.ReadKey()
```

7.- Otros métodos:

- **Clear**, borra la pantalla:

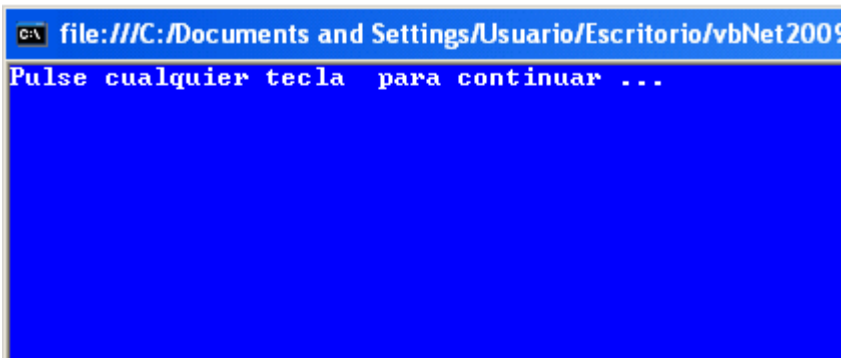
```
    Console.Clear()
```

- **BackgroundColor,** cambia el color de fondo de la consola

```
Console.BackgroundColor = ConsoleColor.Blue
Console.Clear()  'borra la pantalla y el fondo lo pone del color
indicado
```

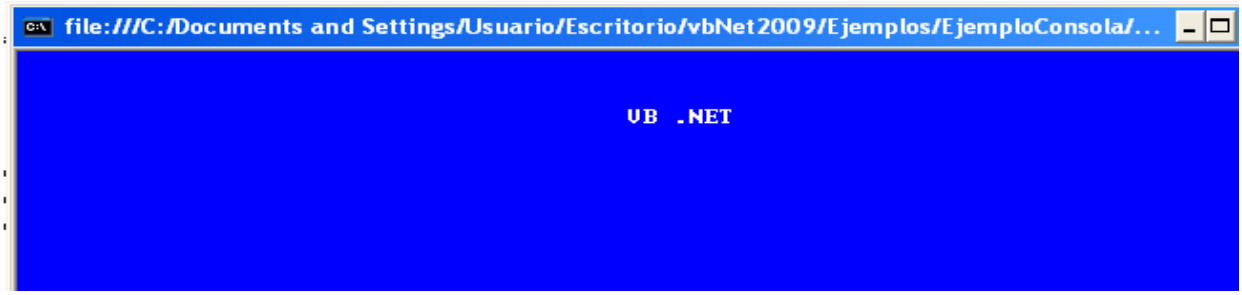
- **ForeGroundColor,** cambia el color de la letra

```
Console.BackgroundColor = ConsoleColor.Blue
Console.Clear()  'borra la pantalla y el fondo lo pone del color
indicado
Console.ForegroundColor = ConsoleColor.White
```



- **SetCursorPosition,** indica en que posición de la pantalla se va a situar el texto.

```
Console.SetCursorPosition(40, 3)
Console.WriteLine("VB .NET")
```



- **ResetColor**, restaura al color inicial

```
Console.ResetColor()
Console.Clear()
```