

## Bloque 2. Visual Basic .NET

## UT 3. Confección de interfaces de usuario y creación de componentes visuales

**TEMA 3****El IDE. Aplicaciones de consola****1. El IDE**

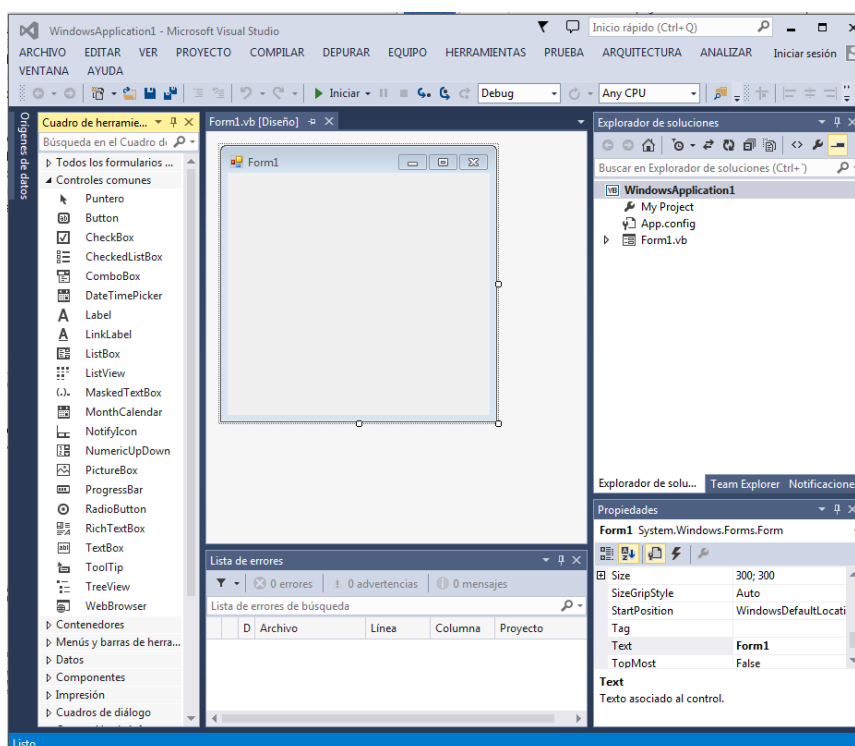
En este tema veremos todas las partes en que se divide el Entorno de Desarrollo Integrado o, a partir de ahora IDE. Cuando conozcamos bien este entorno ganaremos en productividad ya que nos ofrece varias herramientas y ayudas que nos facilitarán mucho nuestro trabajo.

Hasta hace muy poco los lenguajes de Microsoft tenían IDE's diferentes. Ahora no, con .NET llegamos a la total integración, es decir, se dispone de un IDE común.

Ahora sólo hay que iniciar Visual Studio .NET y elegir el lenguaje con el que vamos a trabajar. Además al ser .NET multilenguaje podemos añadir lenguajes de terceros a este entorno. Es habitual en ingeniería que se incorporen kits de desarrollo de determinadas máquinas y que tengan su propio lenguaje y entorno IDE. Si cumplen con el entorno .NET Framework podemos integrarlo en nuestro IDE.

**1.1. El inicio**

Repitamos otra vez el inicio de una aplicación para ver los elementos de esta primera pantalla:

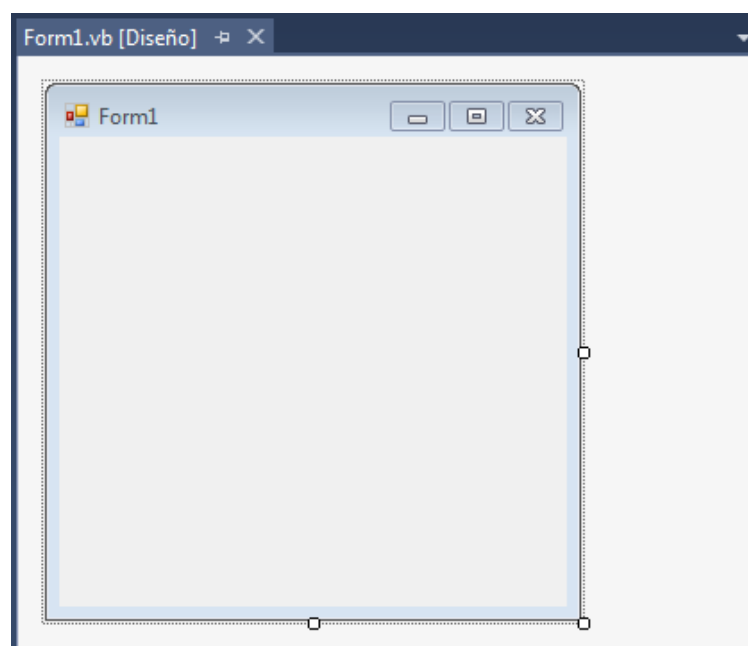


Veamos qué herramientas más importantes tenemos en pantalla:

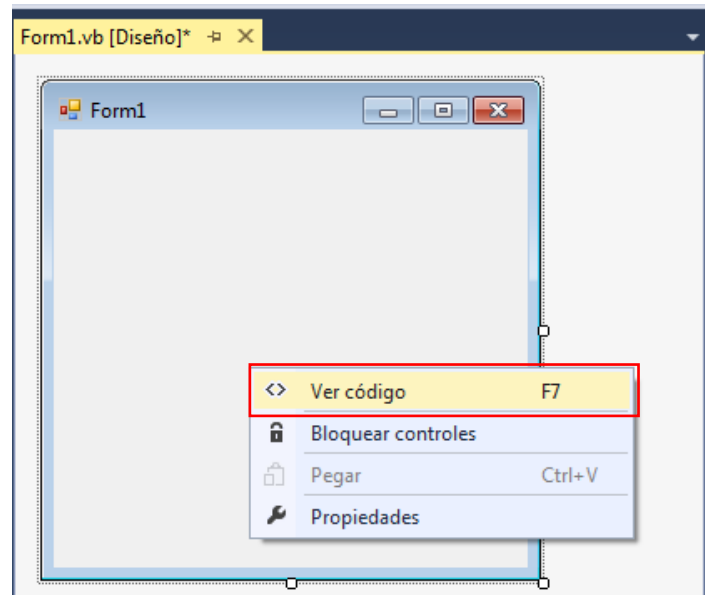
- Parte izquierda. Inicialmente no teníamos desplegado ningún panel, pero vimos cómo dejar visible el cuadro de herramientas haciendo clic en la solapa vertical. Dejaremos visible siempre este panel haciendo clic en la chincheta que aparece en la parte superior del cuadro.
- Cuadro de herramientas. Disponemos de varias secciones con distintas funciones para cada una de ellas. De momento dejamos las predeterminadas. Si las exploramos veremos los grupos de control que tenemos en esta versión: controles comunes, controles contenedores, de menú... La segunda pestaña nos mostrará los orígenes de datos para conectarnos a bases de datos.
- Parte central: formulario de edición. Aquí nos podremos mover con las pestañas superiores por los distintos formularios que tengamos abiertos.
- Parte derecha: son ventanas muy importantes y que tendremos que tener siempre visibles. Tenemos el explorador de soluciones y debajo, la imprescindible ventana de propiedades.
- Parte inferior: por defecto aparece la ventana de resultados pero configuraremos un par de ellas que nos serán de gran ayuda a lo largo del proyecto. Sobre todo durante la ejecución de los programas y poder ver información de depuración.

Cada una de estas partes la veremos con detalle a continuación.

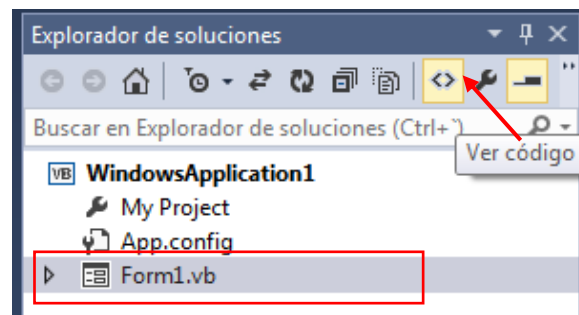
## 1.2. Ventana principal de trabajo



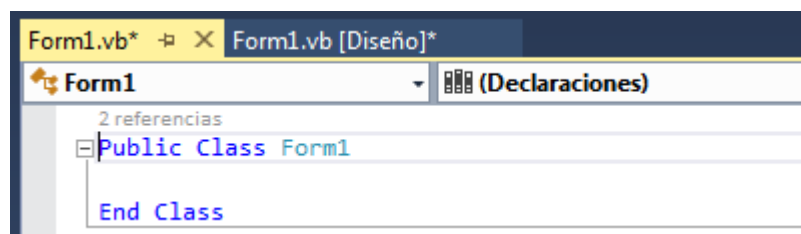
Es la ventana que más área de trabajo utiliza y la que habitualmente utilizaremos para introducir nuestro código. Para pasar a ver el código de este formulario podemos hacer dos cosas. Una, en el formulario elegimos "Ver código" haciendo clic con el botón derecho del ratón:



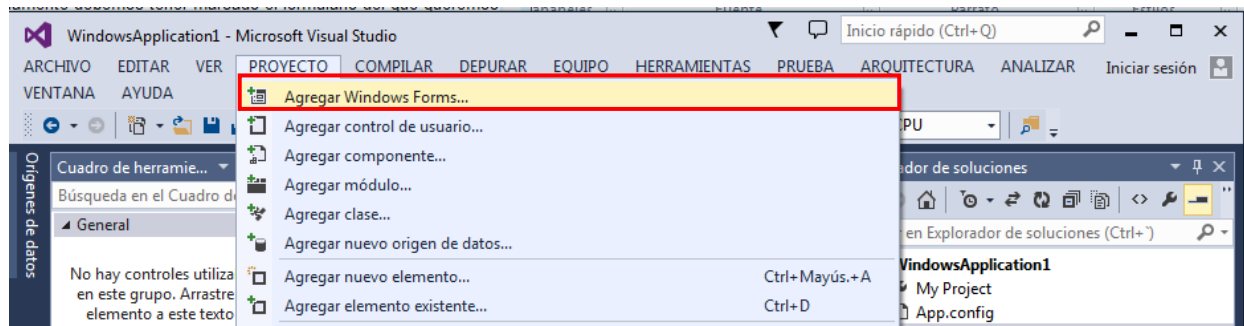
O pulsando en el Explorador de soluciones de la derecha el botón que indica "Ver código":



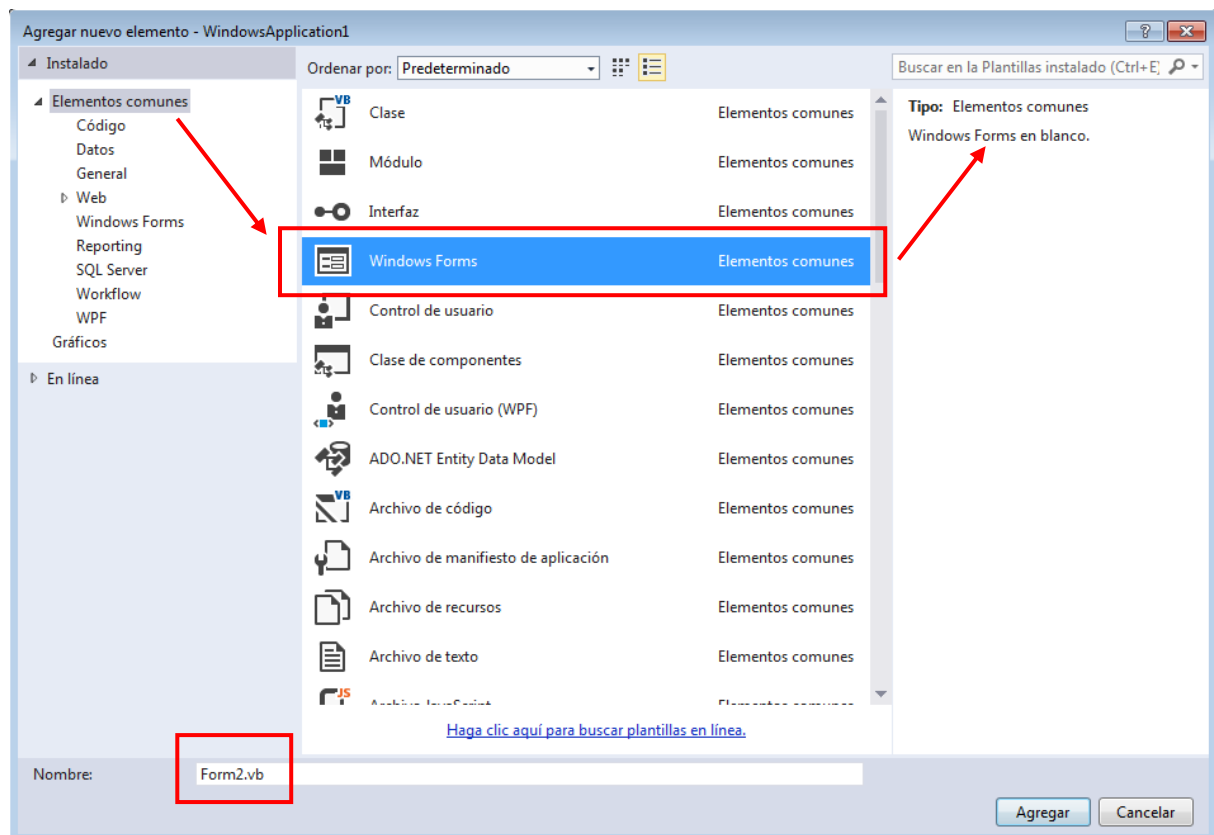
Para llegar a la ventana de código previamente debemos tener marcado el formulario del que queremos ver el código



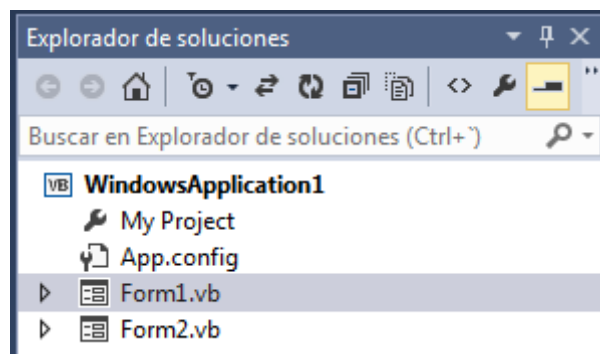
Vamos a añadir un nuevo formulario para ver cómo navegar entre ellos. En el menú "Proyecto" seleccionamos la opción "Agregar Windows Forms" o "Agregar nuevo elemento".



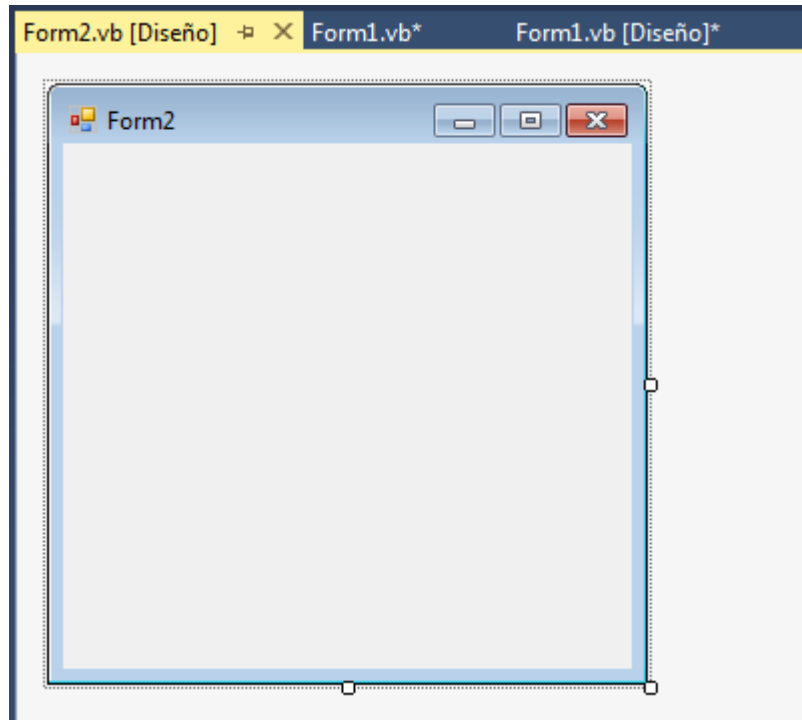
Indicamos que queremos un formulario:



Al pulsar en "Agregar" lo veremos inmediatamente en el explorador de soluciones:



Podemos ver cómo ahora hay una pestaña más con este segundo formulario.



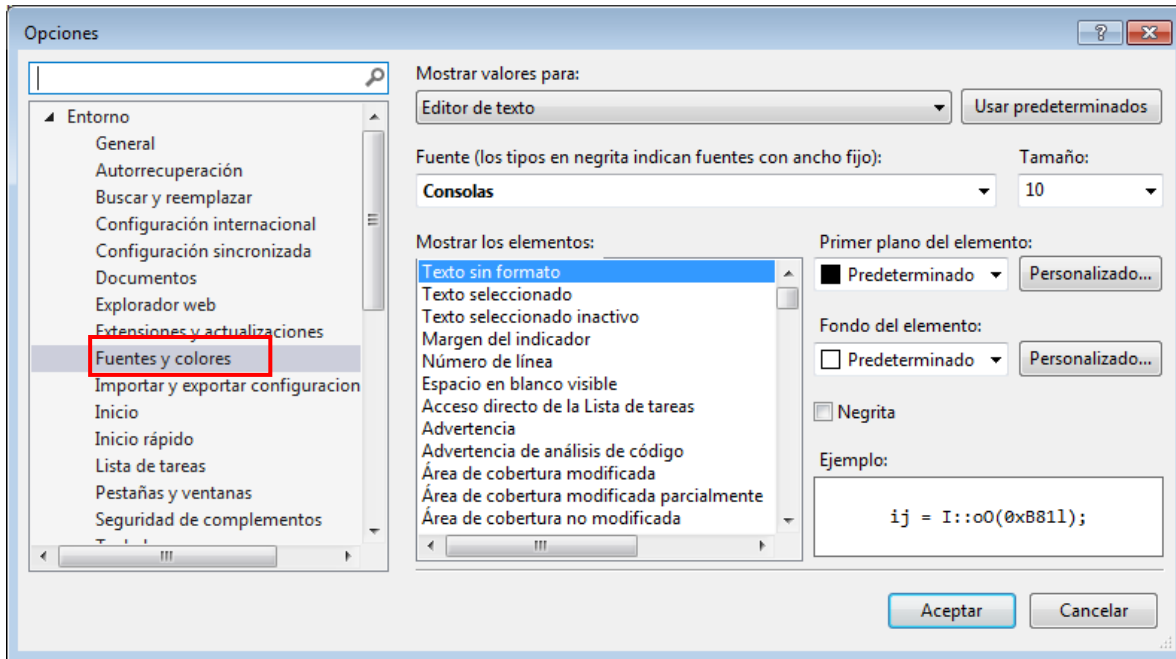
Podemos cambiar con las teclas Control+Tabulador o lógicamente con el ratón. Para cerrar alguna de ellas pulsaremos las teclas Control + F4 o en la "x" de la derecha.

En esta ventana escribiremos el código del programa en forma de declaraciones, procedimientos, funciones, módulos,...

Veamos algunas posibilidades para personalizar esta ventana.

### Ajustar el color y el tipo de letra

Para cambiar el tipo de letra y algunos detalles más del estilo de la fuente seleccionaremos "Opciones" del menú "Herramientas" y luego en la parte del Entorno seleccionamos Fuentes y colores:

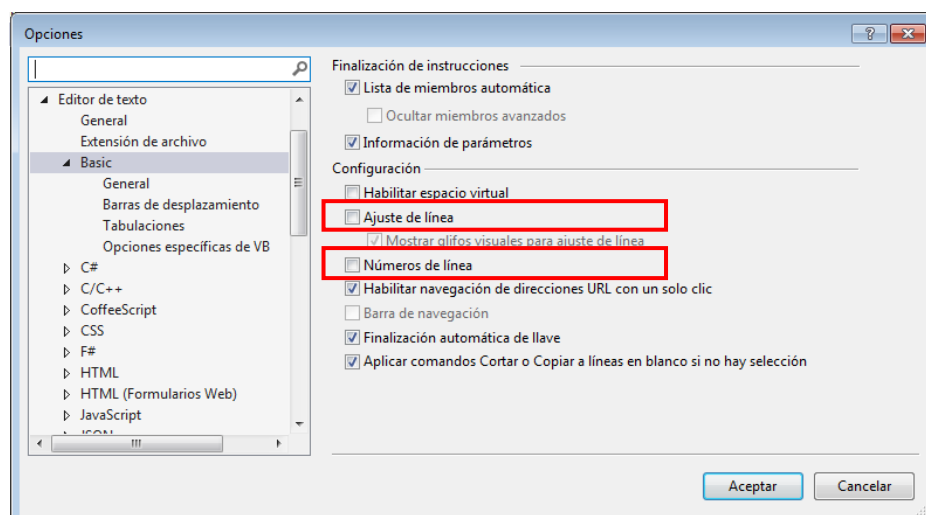


Aquí se pueden modificar cualquiera de los estilos utilizados en el editor, no sólo para la edición, sino, para el comportamiento en general.

**Nota:** Se recomienda que el tipo de letra utilizado sea de paso proporcional como la "Courier" que es la normalmente utilizada. Esta fuente de paso fijo deja el mismo espacio para la letra "i" que para la "m" con lo que las columnas quedan todas alineadas. De no utilizarlo el manejo del código con los sangrados y tabulaciones nos dejará unos listados muy incómodos.

## Mostrar números de líneas

Por defecto esta opción está activada. Para desactivarla utiliza la opción Basic, General de la sección "Editor de texto" de la pantalla anterior:

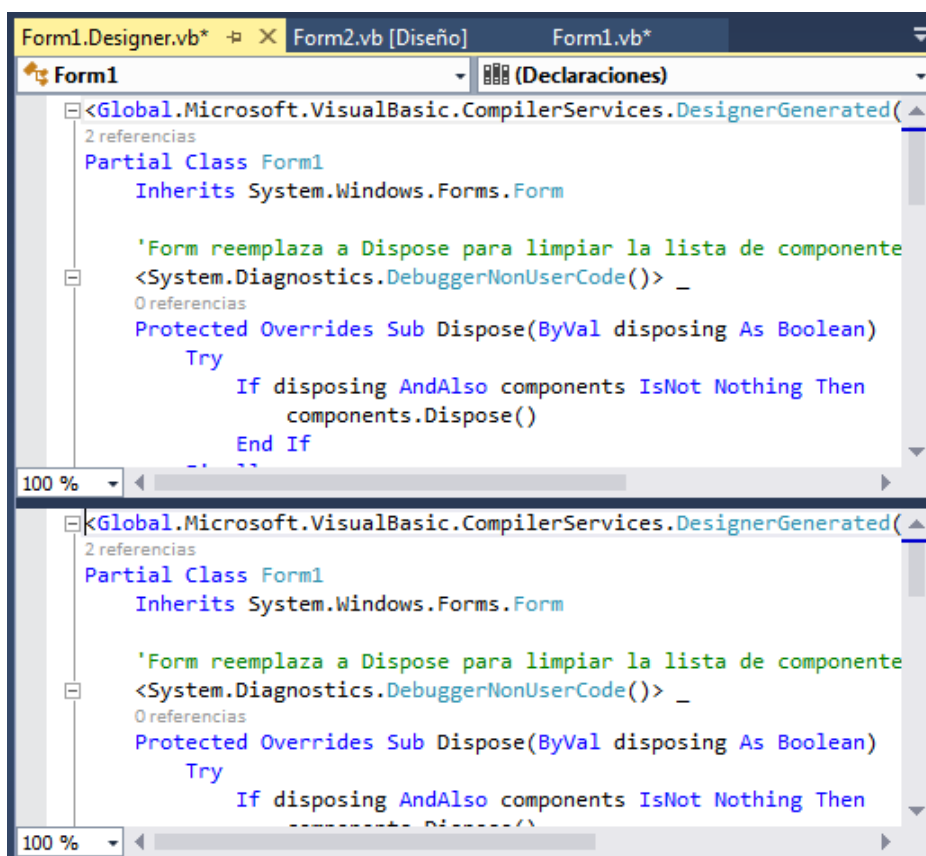


Gracias a esta opción podremos utilizar atajos como los que ofrece Excel o Project de ir hasta una línea en concreto utilizando la opción Ir a del menú *Edición*.

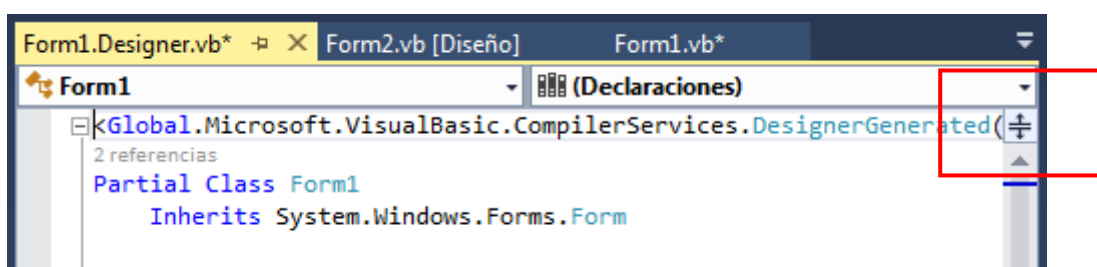
## Dividir la ventana

Al igual que en otros programa de Office podemos dividir la ventana de código. Es especialmente útil si queremos visualizar permanentemente un área de código y queremos ver otra a la vez para, por ejemplo, comparar secciones de código, copiar y pegar,...

Para esto utilizaremos la opción *dividir* del menú *Ventana* o podemos hacer clic en el divisor de ventana que se encuentra justo encima de la barra de desplazamiento:

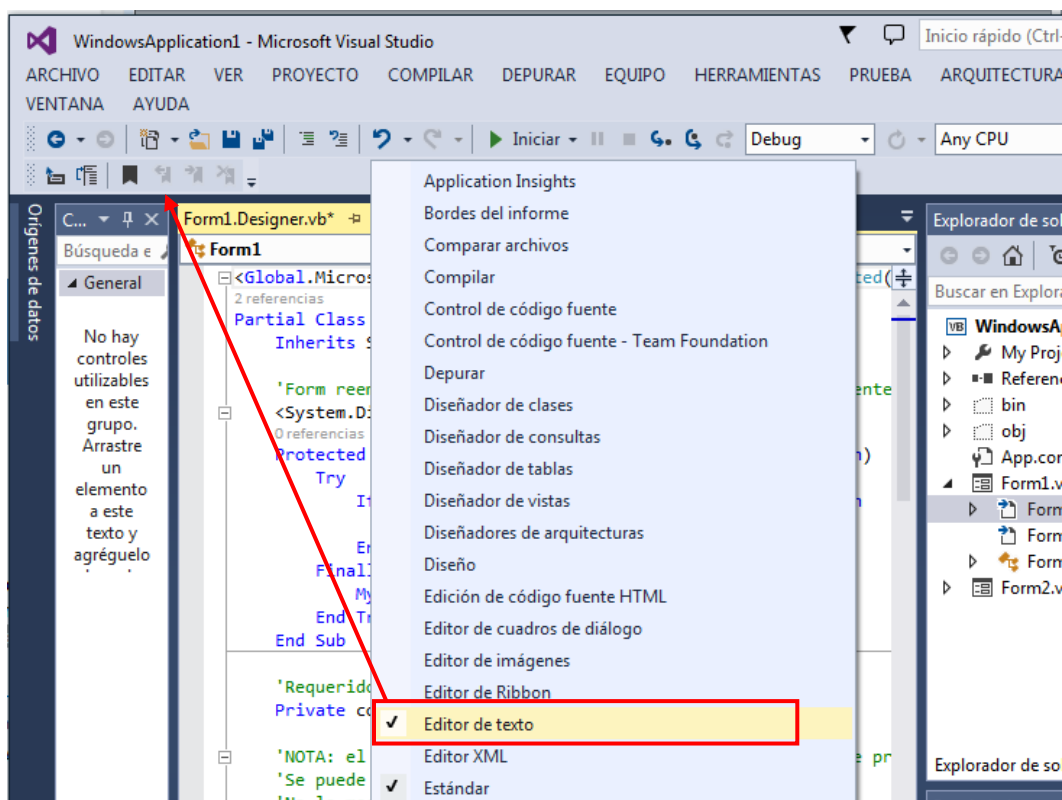


También podemos hacer clic en el divisor de ventana que se encuentra justo encima de la barra de desplazamiento:

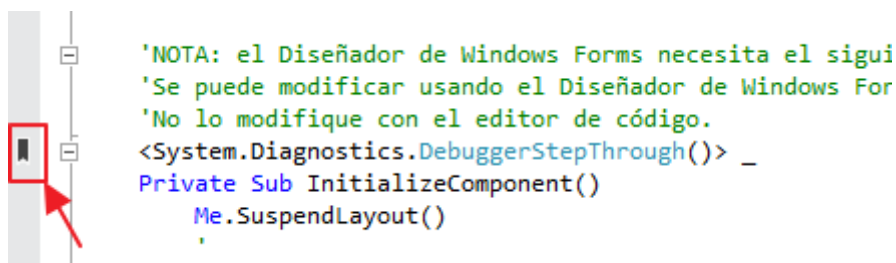


## Marcadores

Los marcadores son señales que introducimos en las líneas de código. Con esto podremos volver rápidamente a la línea en la que colocamos el marcador. Es especialmente útil en listados grandes para ver una determinada parte y volver donde estábamos editando rápidamente. Para insertar un marcador mostraremos primero la barra de botones "Editor de texto" que la obtenemos pulsando con el botón derecho en las barras de botones que tengamos desplegadas:



Los iconos indicados con la flecha son los que controlan estos marcadores, que se representan de esta forma:



Si tenemos más de un marcador nos podemos mover entre ellos desde los iconos de la barra.

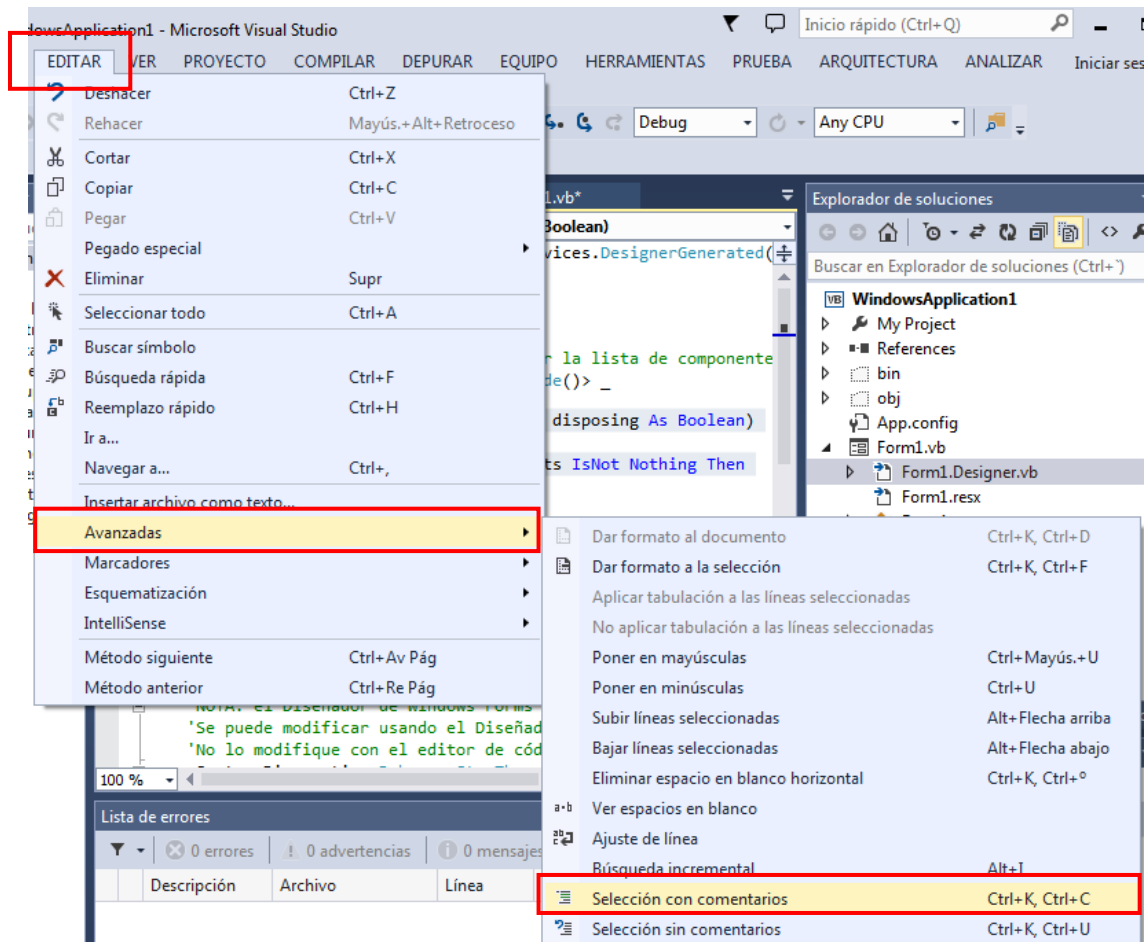




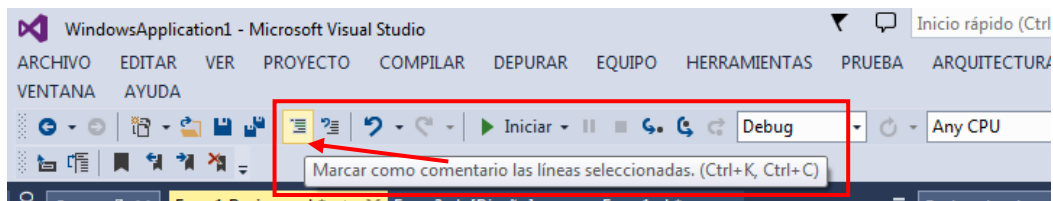
```
Private Sub ListBox1_SelectedIndexChanged(sender As Object, e As _  
    EventArgs) Handles ListBox1.SelectedIndexChanged  
    'Cuando pulsamos en un elemento de la lista...  
    With ListBox1  
        'Obtenemos su contenido con el método GetItemText  
        TextBox1.Text = .GetItemText(.SelectedItem)  
    End With  
End Sub
```

También nos pasará que tengamos que retocar el código de otra persona. Si su código está detallado nos alegraremos de que tenga un buen estilo de programación comentando debidamente el código para poder revisarse con detalle. Otras veces simplemente queremos que no se ejecute un fragmento de código y queremos que se quede momentáneamente como comentarios.

Además de poner los comentarios necesarios a medida que escribimos el código (incluyendo una comilla simple ' antes del comentario) podemos hacer que el editor nos marque una zona como comentario, es decir, sin valor para VB.NET que lo ignorará. Para esto, marcamos las líneas que queramos y utilizamos la opción del menú “Selección con comentarios” de la sección “Avanzadas” en el menú “Editar”.



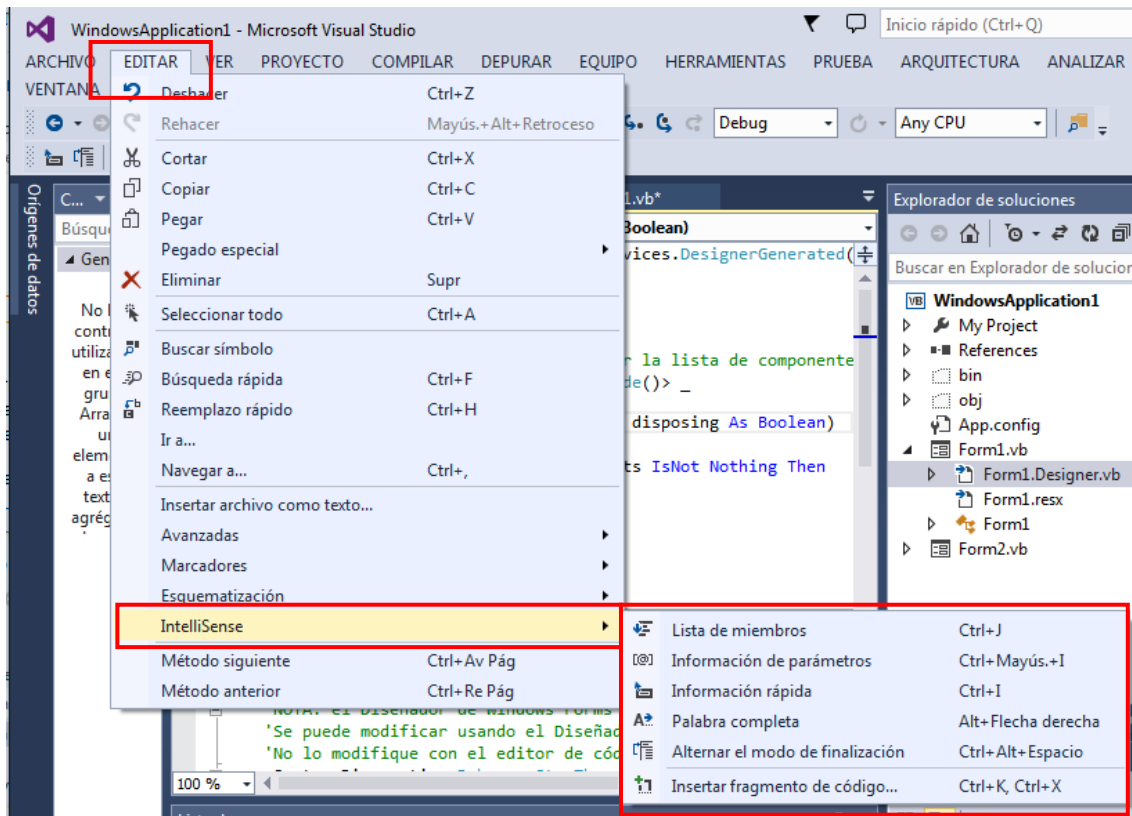
También disponemos de los iconos para realizar esto:



## Intellisense

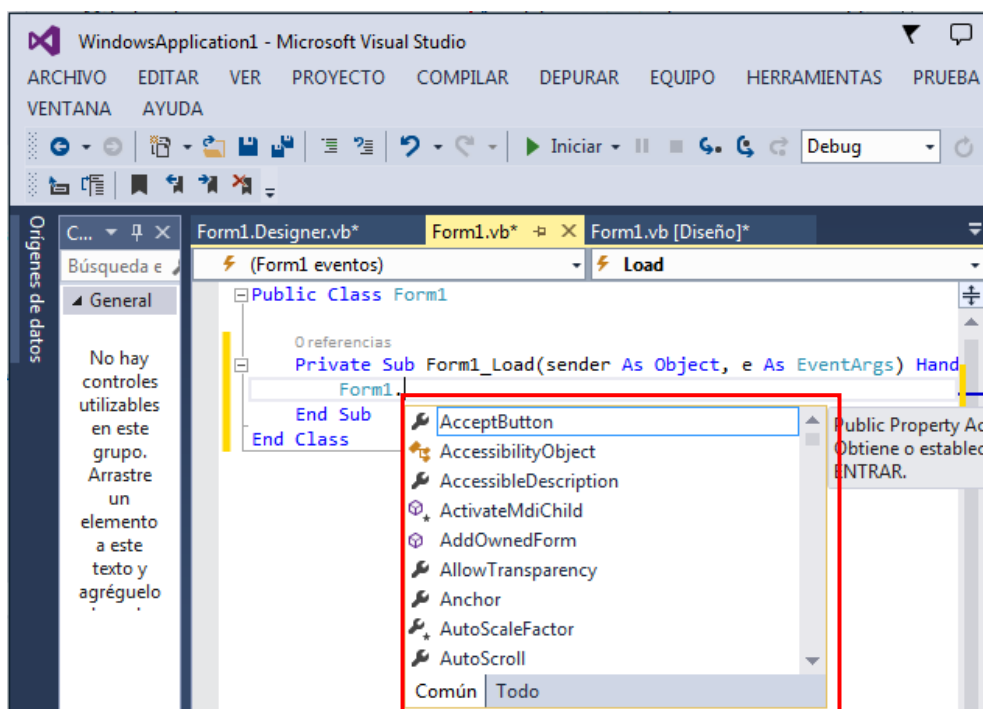
Para terminar con esta parte del editor vamos a ver una tecnología que nos ayudará a escribir el código y, en muchas ocasiones, nos servirá como comprobación de la sintaxis del código que estamos escribiendo.

En el menú "Editar" podemos ver las partes de que se compone este "Intellisense":



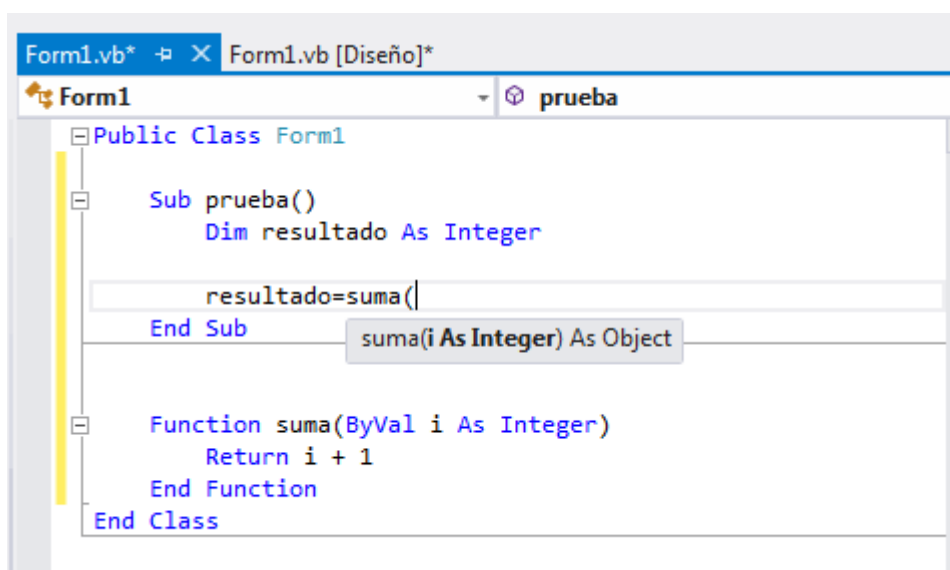
Veamos a qué se refiere cada una de ellas:

- Lista de miembros. Estamos escribiendo el código y estamos trabajando con el objeto formulario. Este formulario tiene muchas propiedades y métodos. Si está bien declarado y escribimos su nombre al escribir el punto "." para asignarle una propiedad o trabajar con un método.

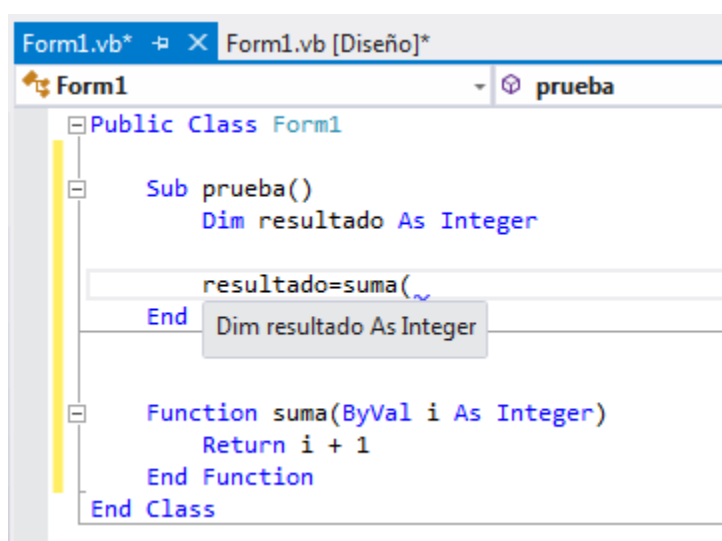


Veremos una lista desplegable con todo lo que podemos hacer o escribir con el objeto "form1". Al seleccionar un elemento de la lista nos confirma que esa instrucción es de "form1"

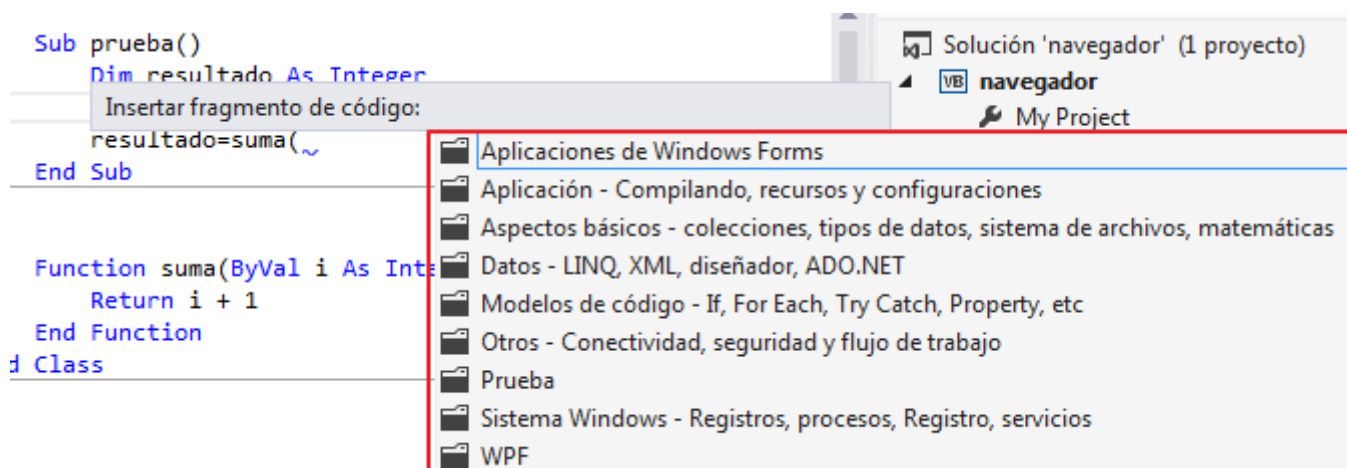
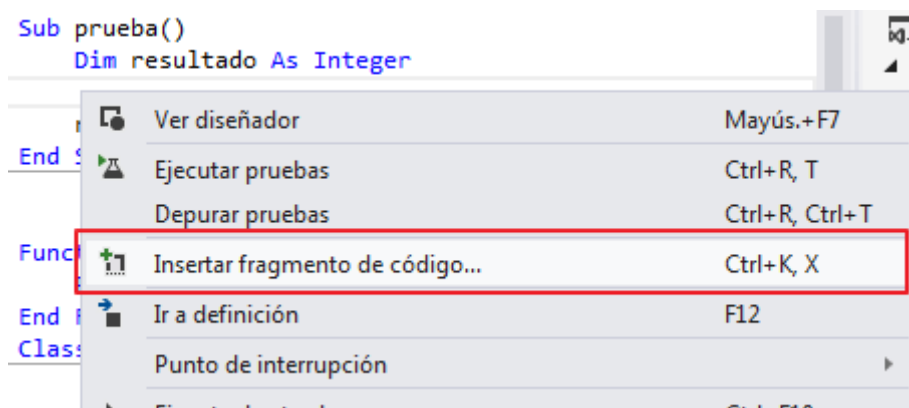
- Información de parámetros. Cuando llamemos a procedimientos y funciones (subprogramas o fragmentos que realizan una operación o proceso) éstos necesitan normalmente unos parámetros para poder trabajar correctamente. Pues bien, en el momento que escribamos la llamada a este procedimiento o función nos irá indicando las variables o parámetros que debemos escribir y de qué tipo. En este ejemplo se llama a una función que necesita un parámetro, al escribir la llamada nos lo indica en la ayuda:



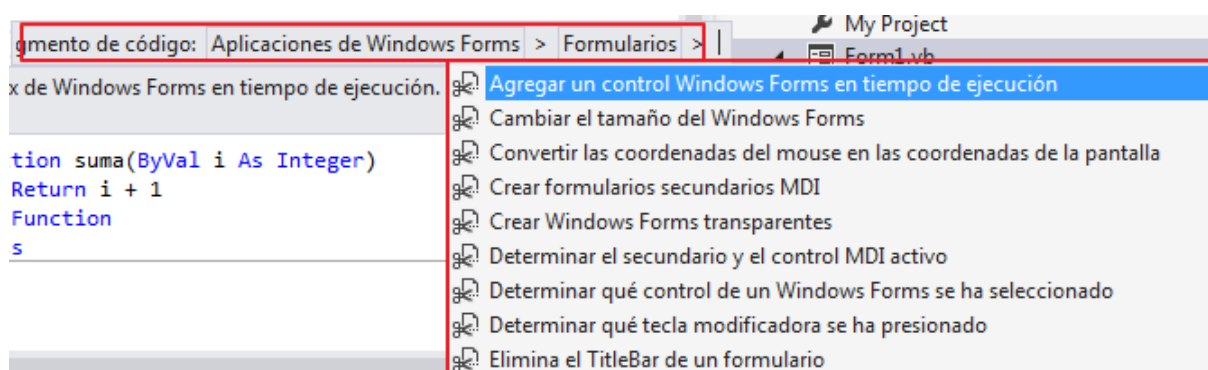
- Información rápida. Nos muestra una viñeta con información.



- Insertar fragmento de código. Con esta opción podemos añadir código genérico que nos servirá de plantilla para seguir escribiendo. Pulsaremos con el botón derecho.



Por ejemplo, seleccionamos código relativo a los formularios. Al indicarlo nos propone estas situaciones:



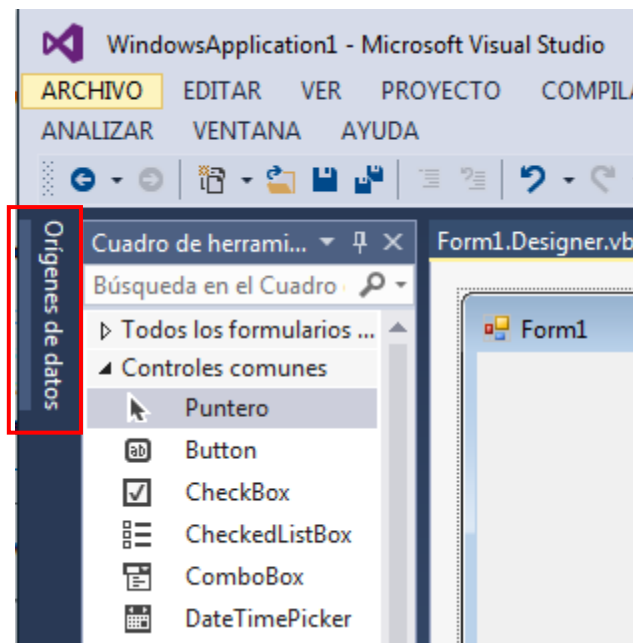
Al seleccionarlas nos introducirá el código correspondiente. Personalmente es más costoso esto que consultar la ayuda, es una ayuda más.

- Palabra completa. Escribe el resto de una variable, un comando o un nombre de función una vez que se han especificado los suficientes caracteres para identificar de forma exclusiva esos elementos.
- Alternar el modo de finalización. Cuando comenzamos a escribir el nombre de un objeto, IntelliSense muestra todos los miembros válidos en una lista desplazable. Al escribirlos, se resalta el símbolo adecuado.

### 1.3. Ventanas adicionales

Además de la ventana principal que hemos visto antes tenemos una serie de ventanas suplementarias que nos ayudarán en nuestro trabajo. Vamos a ver cómo trabajar con ellas para adaptarlas a nuestro gusto.


Las ventanas adicionales están por defecto ocultas en los laterales de nuestro IDE, indicando con una ficha que están ocultas pero disponibles y que se desplegarán al pulsar con el ratón. Esto es verdaderamente útil porque las utilizaremos en algunas ocasiones pero no desearemos que nos ocupen permanentemente un área de pantalla. Veamos este ejemplo:

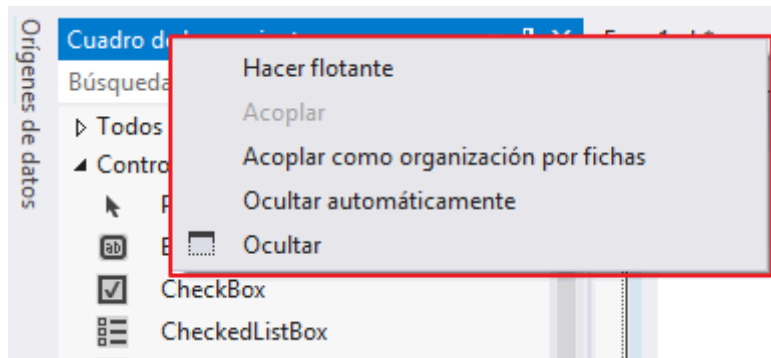


Vemos que está activa la ventana del "Cuadro de Herramientas" y que a la izquierda tenemos disponible la ventana de "Orígenes de datos". Basta con pulsar con el ratón sobre esta pestaña para que se active la ventana. Si en esa área lateral pulsamos con el botón derecho nos aparecerá una lista con todas las ventanas disponibles.

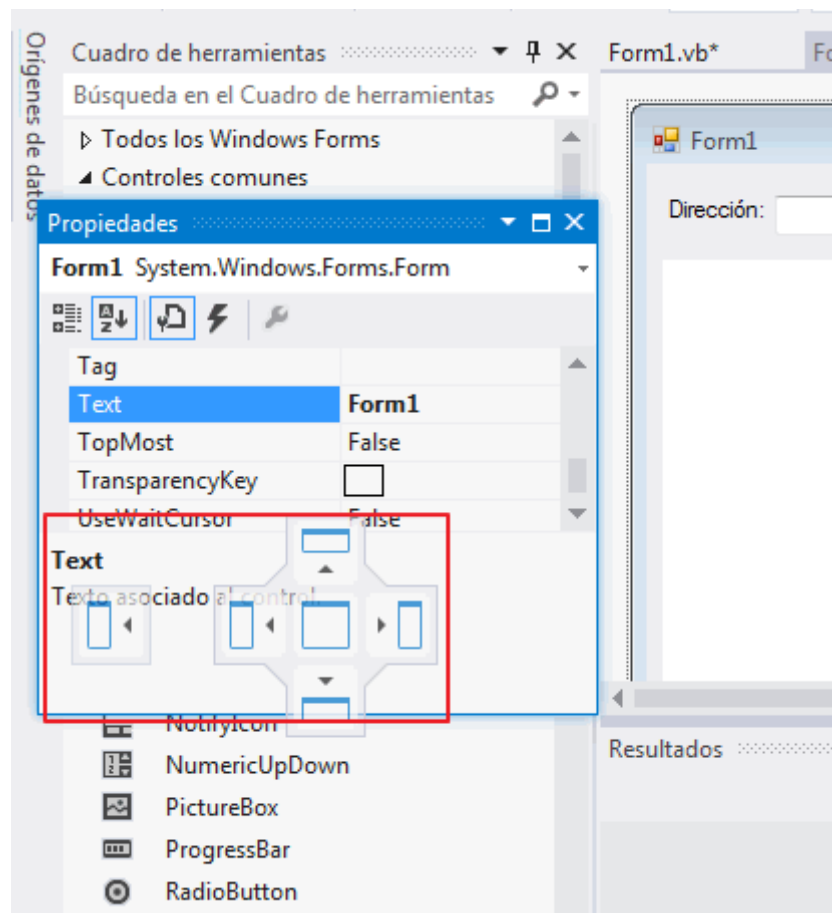
Normalmente estas ventanas disponen de cuatro posibilidades de visualización:

- Acoplable. Se acopla con las ventanas contiguas repartiendo el espacio. Por ejemplo, es habitual tener acopladas en una misma columna la ventana del explorador de soluciones y la ventana de Team Explorer.
- Ocultar. La ventana no se muestra.
- Flotante. Está visible pero no se acopla en ningún sitio. Permanece en primer plano en nuestro IDE.
- Ocultar automáticamente. Es la posición predeterminada y se activa pulsando el ratón en la pestaña.

En ocasiones querremos que una ventana permanezca visible permanentemente. Para esto, una vez la tengamos visible pulsamos en la chincheta que aparece en la parte superior derecha . Para ver todas las opciones pulsaremos con el botón derecho encima del título y elegiremos el estilo que queramos:

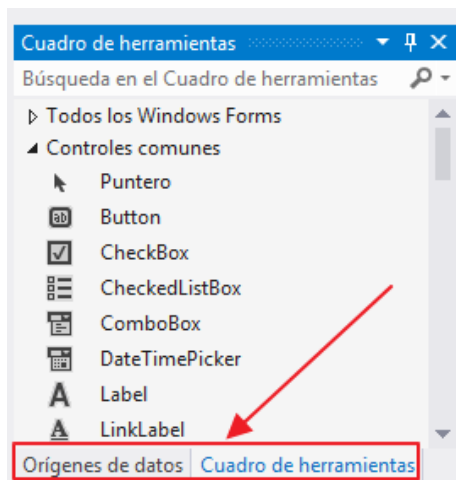


Quizás el más complicado de configurar es el "acoplable" ya que se debe arrastrar con cuidado el título de la ventana para que la acople a la posición que deseemos. Para conseguir que se acoplen varias ventanas arrastraremos una ventana hacia la zona de trabajo de otra y soltaremos el ratón. Si intentamos arrastrar por ejemplo la de propiedades, al moverse aparecerá lo siguiente:



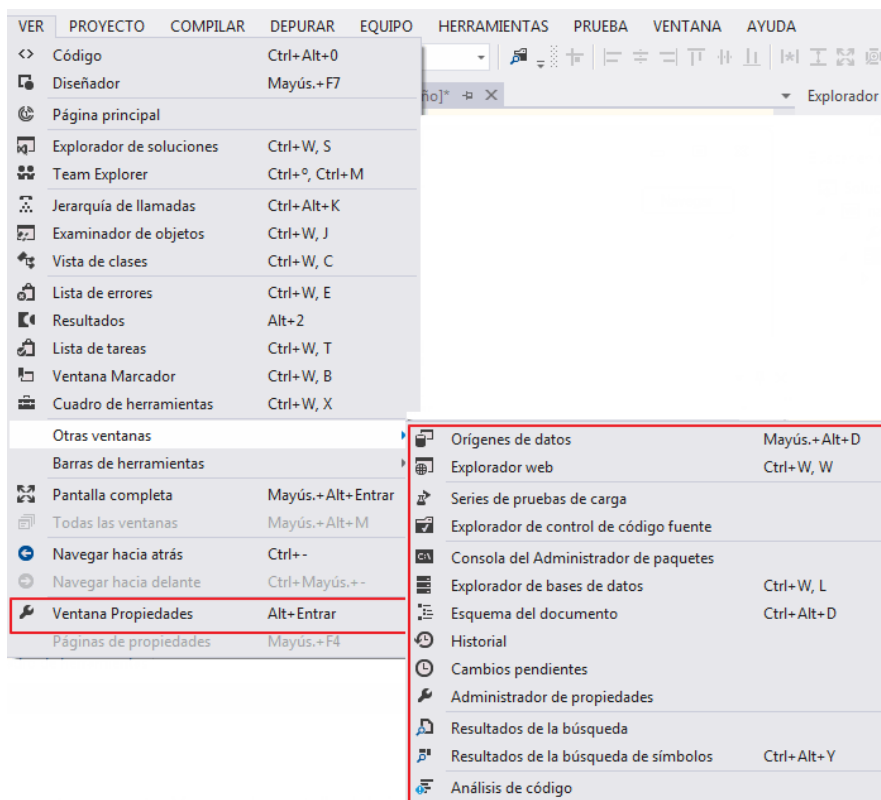
Que son las posiciones donde se acoplará la ventana. En el ejemplo, vemos las posibles direcciones donde quedaría acoplada.

La última posibilidad, y casi la más práctica, es acoplar varias ventanas organizándolas en fichas. De esta forma sólo se ve una y las otras se desplegarán al ir pulsando en los títulos.



Para conseguir acoplarla hemos arrastrado la ventana al cuadro de herramientas que estaba abierto. Si nos fijamos en la parte inferior, están disponibles las ventanas del cuadro de herramientas y el de orígenes de datos.

Si eliminamos accidentalmente alguna ventana, cosa muy probable cuando estamos aprendiendo a manejar el IDE, siempre podemos ir a la opción del menú:



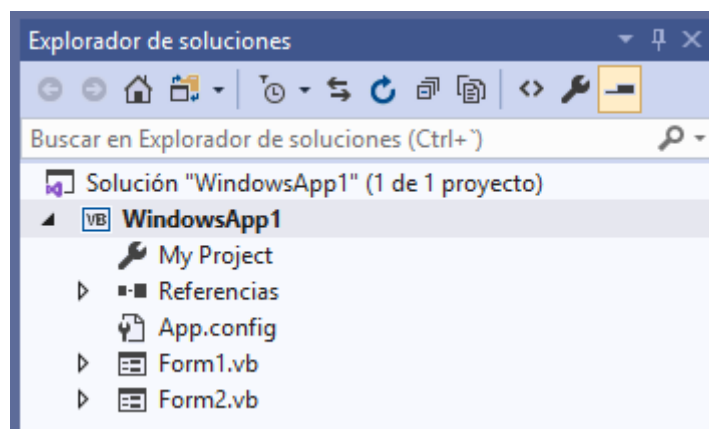
Hay más ventanas que podemos ver en la opción "Otras ventanas" del menú anterior.



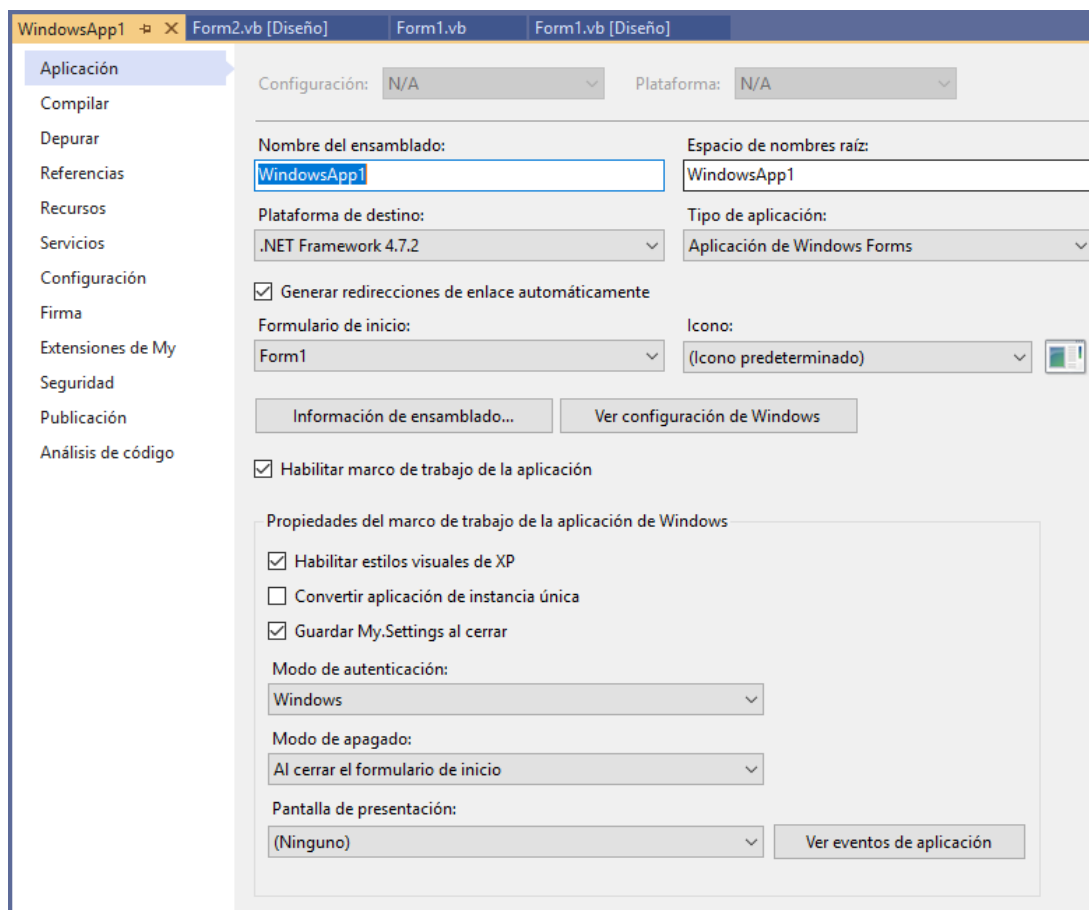
## 1.4. Explorador de soluciones

El explorador de soluciones es una herramienta indispensable. Contiene todos los ficheros de que consta nuestra solución. Permite navegar entre las distintas carpetas y archivos que componen el código fuente de una solución. Una solución puede contener varios proyectos, que se corresponden con varias aplicaciones y/o bibliotecas de clases.

Veamos la pantalla del ejemplo anterior:

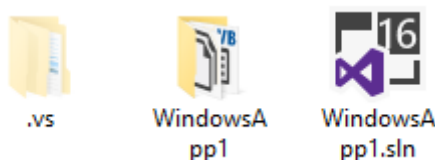


Vemos primero el nombre de la solución y debajo el nombre del proyecto; debajo de él un fichero que pone "My Project" que al hacer doble clic sobre él veremos qué tiene:



Como ves, aparece "Nombre del ensamblado" en lugar de aplicación. Recuerda lo que vimos sobre .NET en el primer tema donde lo que generamos se le llama ahora "ensamblados" aunque lógicamente se trata de una aplicación, pero se utiliza por seguir la nomenclatura de este entorno.

Si vemos la carpeta donde guardamos un proyecto podemos ver estos iconos:



Como ves, hay uno que tiene de extensión ".sln" que es el que define la solución, que en este caso se compone de un proyecto de una sola aplicación de "Windows Forms", o de formularios. En ocasiones será necesario crear una solución con varios proyectos distintos dentro de ella, pero de momento, se compondrán, como hemos hecho hasta ahora, de un solo proyecto.

Volviendo a la ventana de propiedades de nuestro proyecto podemos ver desde el nombre de la aplicación o ensamblado que va a generar hasta el icono que le pondrá al programa cuando lo terminemos. Siguiendo con el explorador de soluciones vemos que aparece el icono del formulario "form1". Si recuerdas, en la carpeta aparecen varios ficheros más, pero se ocultan para tener el área más limpia. Aparece como título el nombre genérico de "WindowsApplicationxx".

### Fichero "App.config"

En el explorador de soluciones aparece un fichero llamado "App.config". Este fichero es similar a los que utiliza el mundo de ASP.NET. Se trata de un fichero XML de configuración en el que podemos definir parámetros de configuración para la aplicación.

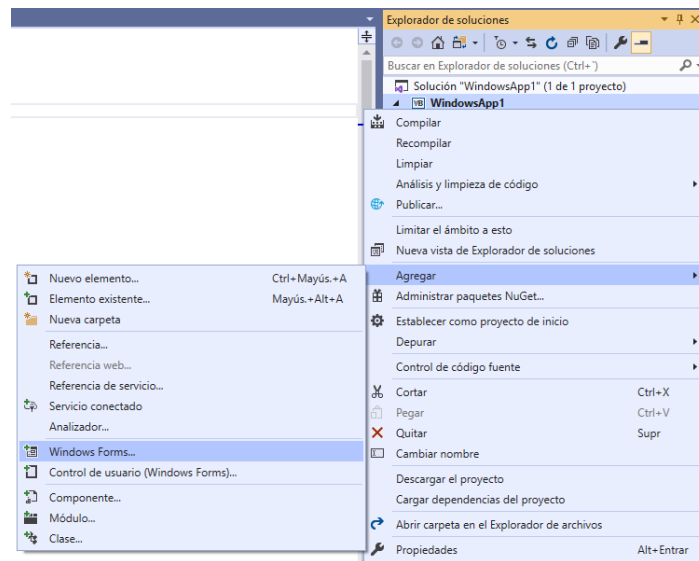
Por ejemplo, cuando definamos una conexión a una base de datos, tendremos que definir una cadena de conexión donde le digamos el servidor y usuario de conexión. En condiciones normales esto lo tendríamos que definir dentro de los formularios. Con este fichero se pretende no escribir ninguna configuración dentro del código VB. Así que definiremos dentro de él los parámetros generales del programa.

Utilizaremos este fichero cuando tratemos con bases de datos.

## 1.5. Añadir más elementos a un proyecto

A lo largo de nuestro proyecto tendremos que añadir más elementos, en su mayoría nuevos formularios. Para esto elegiremos la opción "Agregar Windows Forms" o "Agregar nuevo elemento" del menú Proyecto.

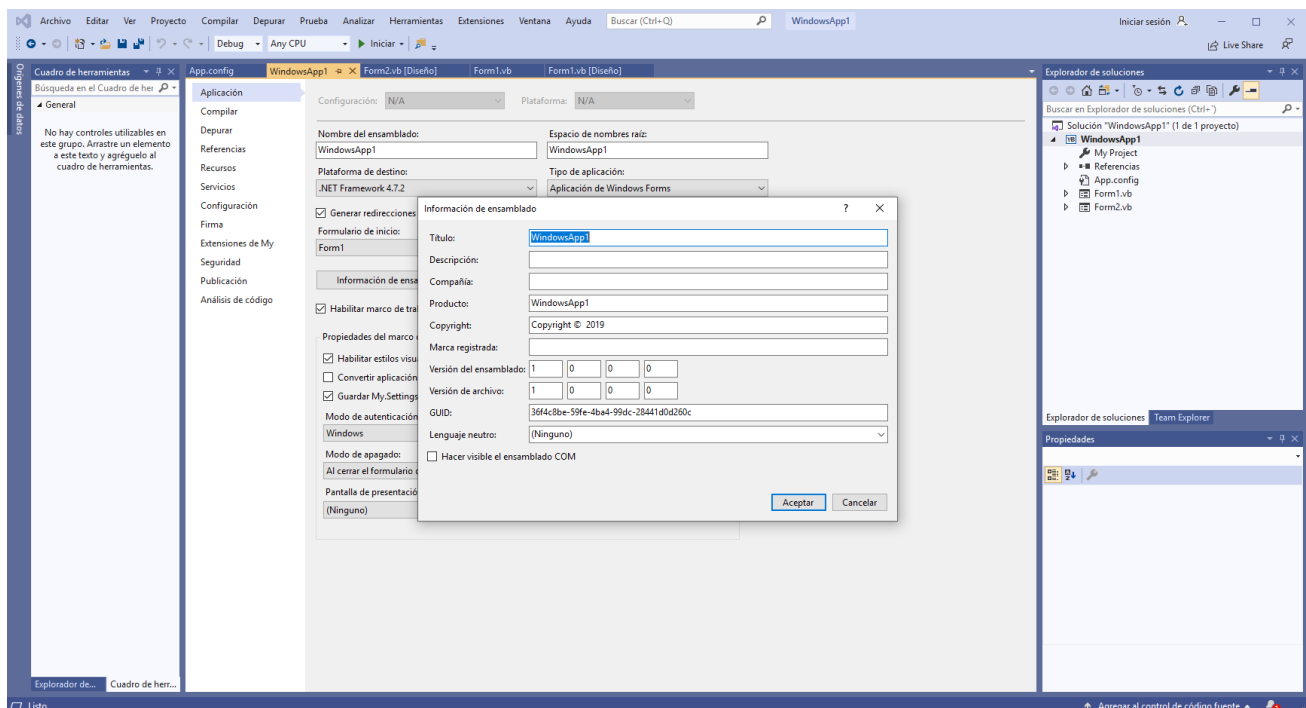
Estas operaciones también se pueden realizar con el botón derecho sobre el nombre del proyecto en el Explorador de soluciones.



## 1.6. Propiedades de la solución y del proyecto

En algunas ocasiones necesitaremos cambiar propiedades del proyecto o de la solución. Para acceder a esta ventana pulsaremos con el botón derecho en la ventana del explorador de soluciones y luego en el título del proyecto o en el título de la solución:

### Aplicación



Como podemos ver, podemos indicarle el nombre del programa (ensamblado), el espacio de nombres, tipo de aplicación e icono. En la pantalla destacada "Información de ensamblado" podemos ver detalles que luego podrá ver el usuario como la versión o autor.

En "formulario de inicio" le podemos indicar cuál de todos los formularios de que consta nuestro proyecto, es el que debe iniciar primero. En la parte inferior tenemos algunos detalles del comportamiento del programa, como por ejemplo, cómo debe salir del programa: cuando se termine el formulario inicial o hasta que se cierre el último formulario.

## Compilar

En la sección "Compilar" podemos ver estas opciones:

Aplicación

Compilar

Depurar

Referencias

Recursos

Servicios

Configuración

Firma

Extensiones de My

Seguridad

Publicación

Análisis de código

Configuración: Activa (Debug) Plataforma: Activa (Any CPU)

Ruta de acceso de salida de la compilación:

bin\Debug\

Opciones de compilación:

Option explicit: On Opción strict: Off

Option compare: Binary Opción infer: On

CPU de destino: AnyCPU

☒ Preferencia de 32 bits

Configuraciones de advertencias:

Condición	Notificación
Conversión implícita	Ninguno
Enlace en tiempo de ejecución; la llamada podría fallar en tiempo de ejecución	Ninguno
Tipo implícito; se supone el objeto	Ninguno
Uso de variable anterior a la asignación	Advertencia
Función que devuelve el tipo de referencia sin valor devuelto	Advertencia
Función que devuelve el tipo de valor intrínseco sin valor devuelto	Advertencia
Variable local no usada	Advertencia
La variable de instancia obtiene acceso al miembro compartido	Advertencia
Acceso a la propiedad u operador de forma recursiva	Advertencia
Bloques catch duplicados o superpuestos	Advertencia

☐ Deshabilitar todas las advertencias

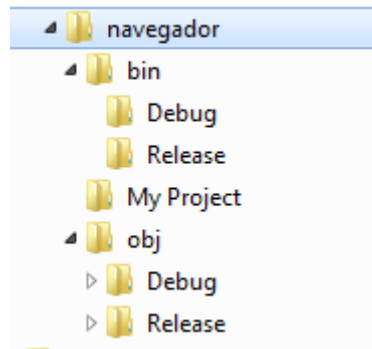
☐ Considerar todas las advertencias como errores

☒ Generar archivo de documentación XML

☐ Registrar para interoperabilidad COM

Opciones avanzadas de compilación...

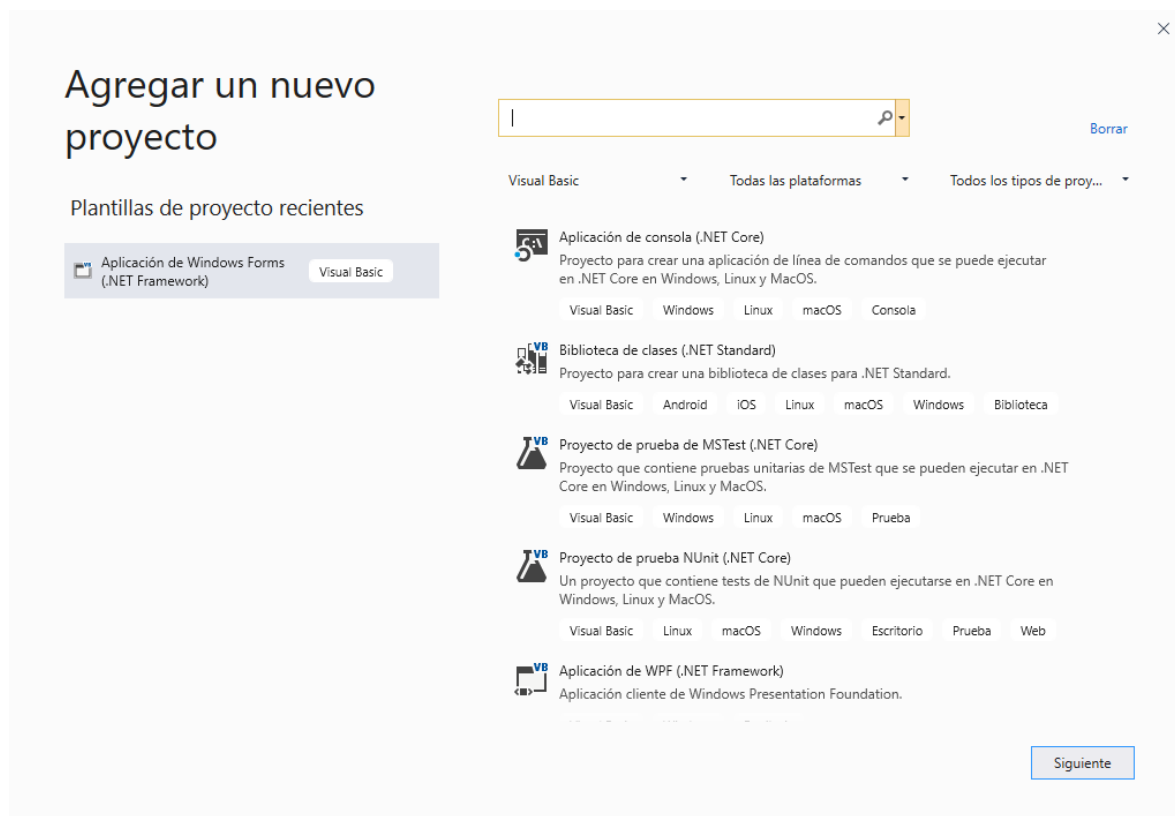
Es la forma en que el compilador generará nuestro programa. Los valores de "Option" los veremos más adelante. Debajo podemos ver qué tipo de acción hará cuando se produzca uno de estos sucesos en la compilación. Por ejemplo, si no hemos utilizado una variable declarada nos mostrará una advertencia. Pero como no es importante, podríamos decirle que no nos lo notifique seleccionando esta opción del cuadro desplegable que tiene a la derecha. La carpeta que aparece en primer lugar, es la carpeta donde nos va a generar la aplicación:



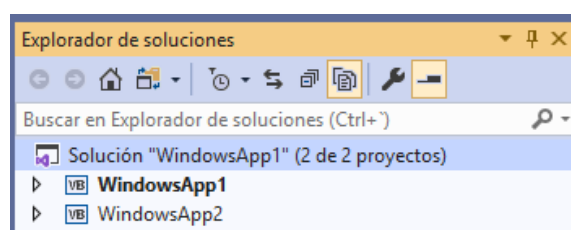
Es decir, la versión final estará en la carpeta "Release" y la versión provisional de depuración, estará en la carpeta "Debug".

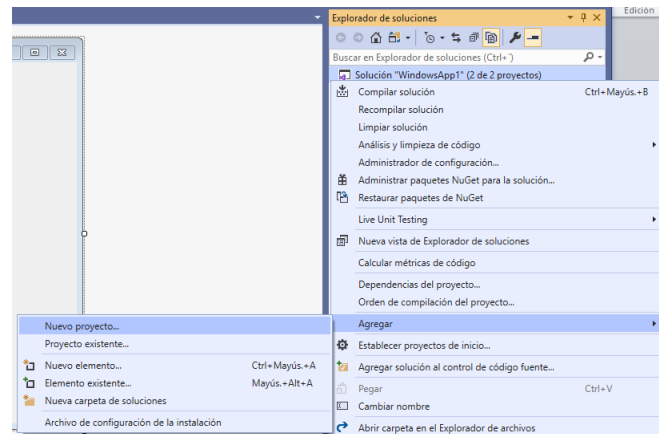
## 1.7. Añadir proyectos a una solución

Antes vimos cómo añadir elementos como formularios y otros a un proyecto. Ahora lo que quiero es añadir otro proyecto a mi solución. Tendremos que guardar el proyecto actual y en el menú *Archivo*, opción *Agregar* → *Nuevo Proyecto*. En este caso la pantalla será como la inicial:



El resultado es el siguiente:

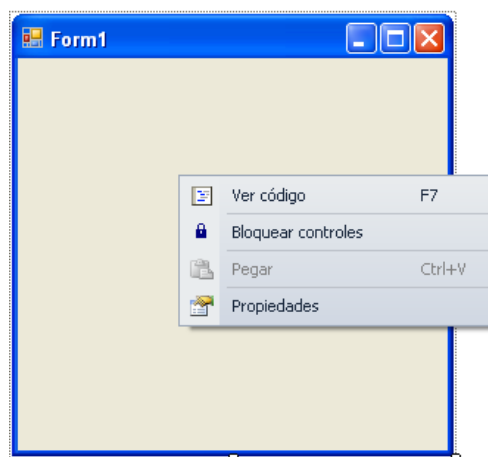




También podemos llegar al mismo resultado del siguiente modo: utilizamos el botón derecho del ratón en el título de la solución del Explorador de soluciones. → Agregar → Nuevo Proyecto o Proyecto Existente

## 1.8. Menú contextual

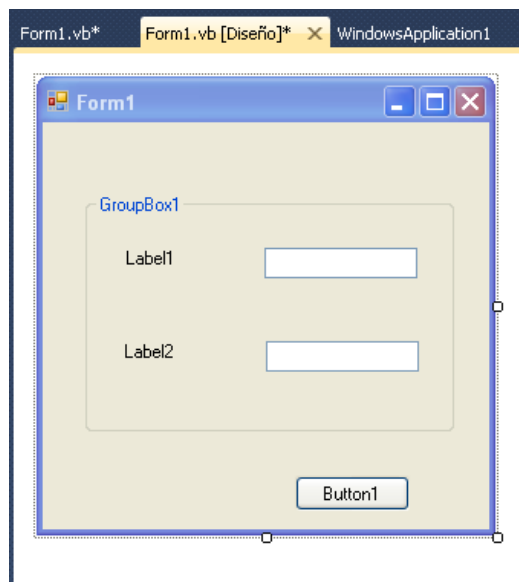
Cuando pulsamos el botón derecho del ratón se despliega un menú que varía según en la situación en la que estemos: ventana de trabajo, explorador. A este menú se le llama contextual porque depende del contexto en el que nos encontremos. Por ejemplo, en el diseño tendremos este menú:



## 1.9. Diseñador del formulario

Esta es una de las áreas más entretenidas y difíciles de manejar. Entretenida porque es la que vamos a utilizar para dibujar nuestras pantallas: seleccionar iconos, controles y otros elementos para construir las pantallas con las que interactuarán los usuarios y difícil porque es difícil diseñar una buena interfaz.

En esta pantalla colocaremos pues los controles que vayamos a necesitar en nuestra interfaz pero en ocasiones utilizaremos controles que no ofrecen una funcionalidad con el usuario pero que necesito en el proyecto: estos controles son ocultos para nuestro proyecto. Por ejemplo un menú, un temporizador, un cuadro de diálogo... los tendremos disponibles para utilizar pero no ofrecen interfaz con el usuario:



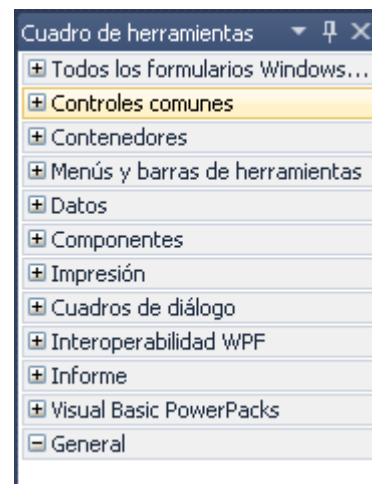
## 1.10. Cuadro de herramientas

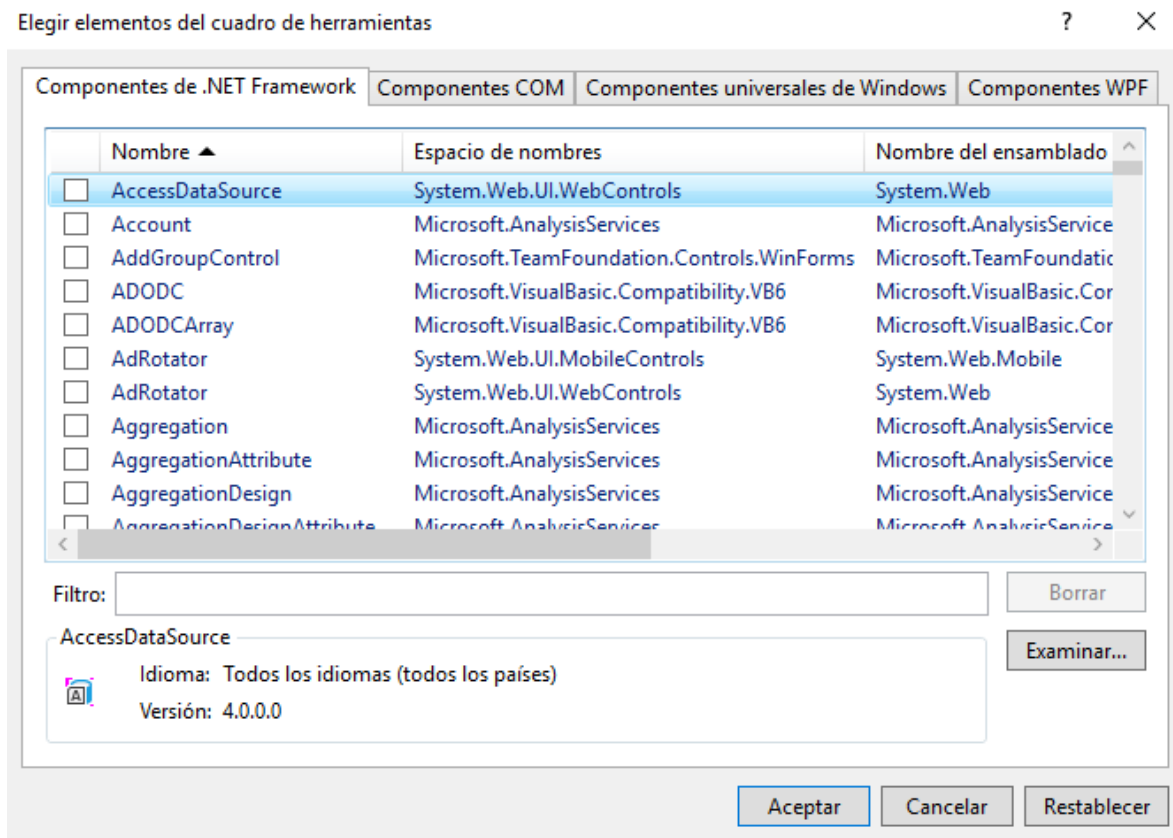
El cuadro de herramientas contiene todos los elementos que podremos incorporar a nuestros proyectos. Como son muchos está dividido en fichas similares a otros programas de Office:

Por defecto, tendremos formularios Windows que son los controles para los formularios. Además de estas fichas tenemos otras disponibles pero que no están visibles. Para que sean visibles pulsaremos con el botón derecho y seleccionamos "Mostrar todos".

Esta pantalla es muy flexible y permite mover de sitio las pestañas simplemente pulsando en el título y arrastrándolo a la posición que queramos.

Por si fuera poco podemos crear nuevas pestañas. Sólo tenemos que seleccionar la opción "Agregar pestaña" que aparece al pulsar con el botón derecho encima de una pestaña. Luego le asignamos un nombre. A continuación pulsamos con el botón derecho sobre ella y seleccionamos "Elegir elementos". Aparecerá la siguiente pantalla que nos permitirá añadir los elementos que queramos.



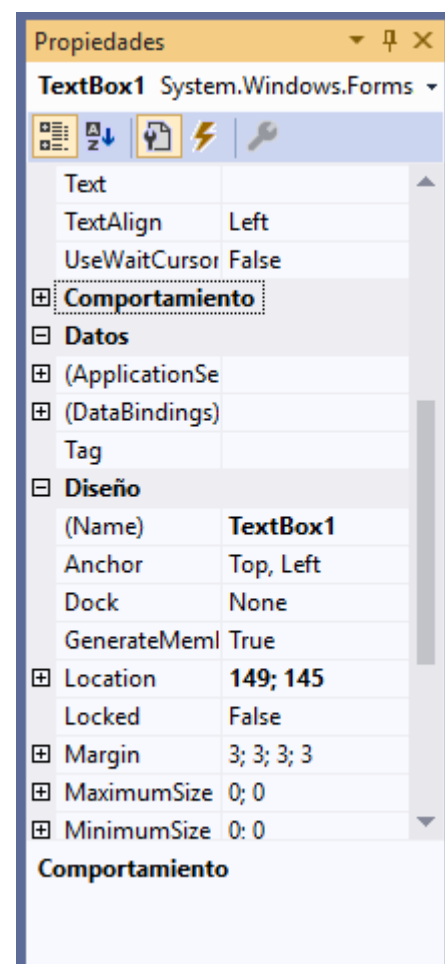


## 1.11. Ventana propiedades

Esta ventana es junto con las dos anteriores las que más vamos utilizar en el diseño ya que aquí es donde daremos nombre a los controles, modificaremos su aspecto y configuraremos su funcionamiento.

Se divide en varias partes. En la parte superior tenemos el nombre del objeto o control: Textbox1 y a la derecha le indica de que tipo es o su clase. Puedes ver que aparece otra vez el namespace "System.Windows.Forms". Así que ya sabemos que los cuadros de texto como este dependen de este digamos superobjeto que es el que tiene todos los controles de Windows.

A continuación tiene una barra de herramientas que nos permite distribuir las propiedades (en ocasiones serán muchas) en una jerarquía, según el tipo de propiedad que es o en orden alfabético.





Para introducir datos según el tipo de propiedad lo podremos hacer de una forma u otra: un menú desplegable, una selección de fichero, una pantalla adicional con editor de texto, un cuadro de diálogo de selección de color...

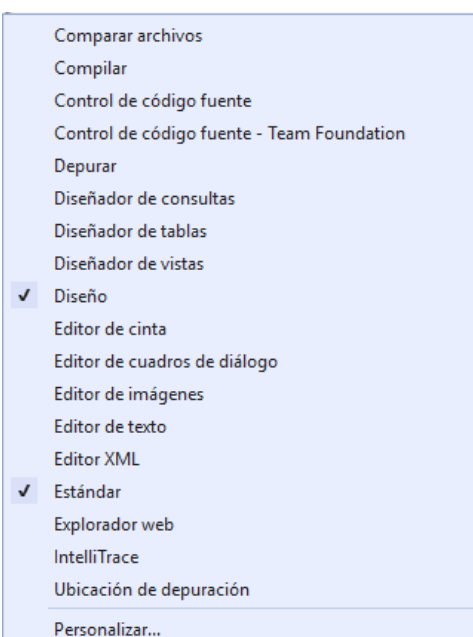
Por último, comentar que en la parte donde tenemos el nombre del control en realidad es un cuadro desplegable que contiene todos los controles de nuestro formulario. En ocasiones el control que queremos modificar puede no estar visible, con esta opción podemos buscarlo en la lista para cambiar alguna propiedad.

## 1.12. Barras de herramientas

En Visual Studio disponemos de multitud de barras de herramientas. Su activación y manejo es igual que el resto de las aplicaciones de Office: bien con el botón derecho en una barra de herramientas o seleccionando el menú Ver y luego Barras de Herramientas:

Con la última opción, Personalizar, podemos crear y modificar barras de herramientas para poner las opciones que nos interesen. En general veremos las más utilizadas o las que nos hagan falta según lo que estemos manejando.

Igual que en las ventanas anteriores las barras de herramientas también se pueden dejar acopladas o flotantes. Para esto haz clic y arrastra la barra utilizando la barra vertical que tienen justo a la izquierda.



## 1.13. Lista de tareas

A medida que escribamos una aplicación nos saldrán pequeños trabajos o tareas que debemos escribir o apuntar y quedarán pendientes. Para esto Visual Studio incluye una sección llamada *Lista de tareas* que permite definir trabajos pendientes de realizar en el código de la aplicación, asignándoles una prioridad.

Podemos verla en Ver → Lista de tareas.

Se usa la **Lista de tareas** para hacer un seguimiento de los comentarios de código que usan tokens como TODO y HACK (o tokens personalizados) y administrar los accesos directos que le

llevarán directamente a una ubicación predefinida del código. Haremos clic en el elemento de la lista para ir a su ubicación en el código fuente.

Cuando la ventana **Lista de tareas** está abierta, aparece en la parte inferior de la ventana de la aplicación.

Para cambiar el criterio de ordenación de la lista, seleccionamos el encabezado de cualquier columna. Para refinar los resultados de la búsqueda, presiona la tecla **Mayúsculas** y elige un segundo encabezado de columna. Como alternativa, elige **Ordenar por** en el menú contextual y elige un encabezado. Para refinar los resultados de la búsqueda, presiona la tecla **Mayúsculas** y elige un segundo encabezado de columna.

Para mostrar u ocultar columnas, elige **Mostrar columnas** en el menú contextual. Selecciona las columnas que quieres mostrar u ocultar.

Para cambiar el orden de las columnas, arrastra cualquier encabezado de columna a la ubicación que quieras.

También aparece un comentario en el código precedido de un marcador de comentario y un token predefinido en la ventana **Lista de tareas**. Por ejemplo, el siguiente comentario de C# tiene tres partes distintas:

- El marcador de comentario (//)
- El token, por ejemplo (TODO)
- El comentario (el resto del texto)

C#

```
// TODO: Load state from previously suspended application
```

Dado que `TODO` es un token predefinido, este comentario aparece como una tarea `TODO` en la lista.

Los tokens predeterminados solo están disponibles para los lenguajes C/C++, C# y VB.

De manera predeterminada, Visual Studio incluye los tokens siguientes: HACK, TODO, UNDONE y UnresolvedMergeConflict. No distinguen mayúsculas de minúsculas. También se pueden crear tokens propios personalizados.

Para crear un token personalizado:

1. En el menú **Herramientas** → **Opciones**.
2. Abre la carpeta **Entorno** y, a continuación, elige **Lista de tareas**.
3. En el cuadro de texto **Nombre**, escribe el nombre del token, por ejemplo, **BUG**.

4. En la lista desplegable **Prioridad**, elige una prioridad predeterminada para el nuevo token.
5. Haz clic en **Agregar**.

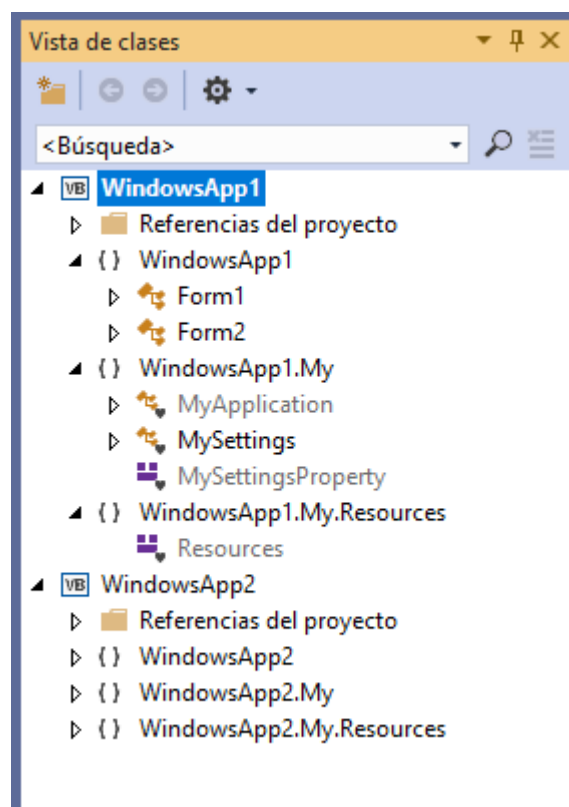
El botón **Agregar** se habilita una vez que especifica un nombre. Debes escribir un nombre antes de hacer clic en **Agregar**.

## 1.14. La vista de clases

En un lenguaje de programación necesitaremos una herramienta que nos permita ver las clases de un objeto. Para activarla podemos utilizar la opción *Vista de clases* del menú *Ver*.

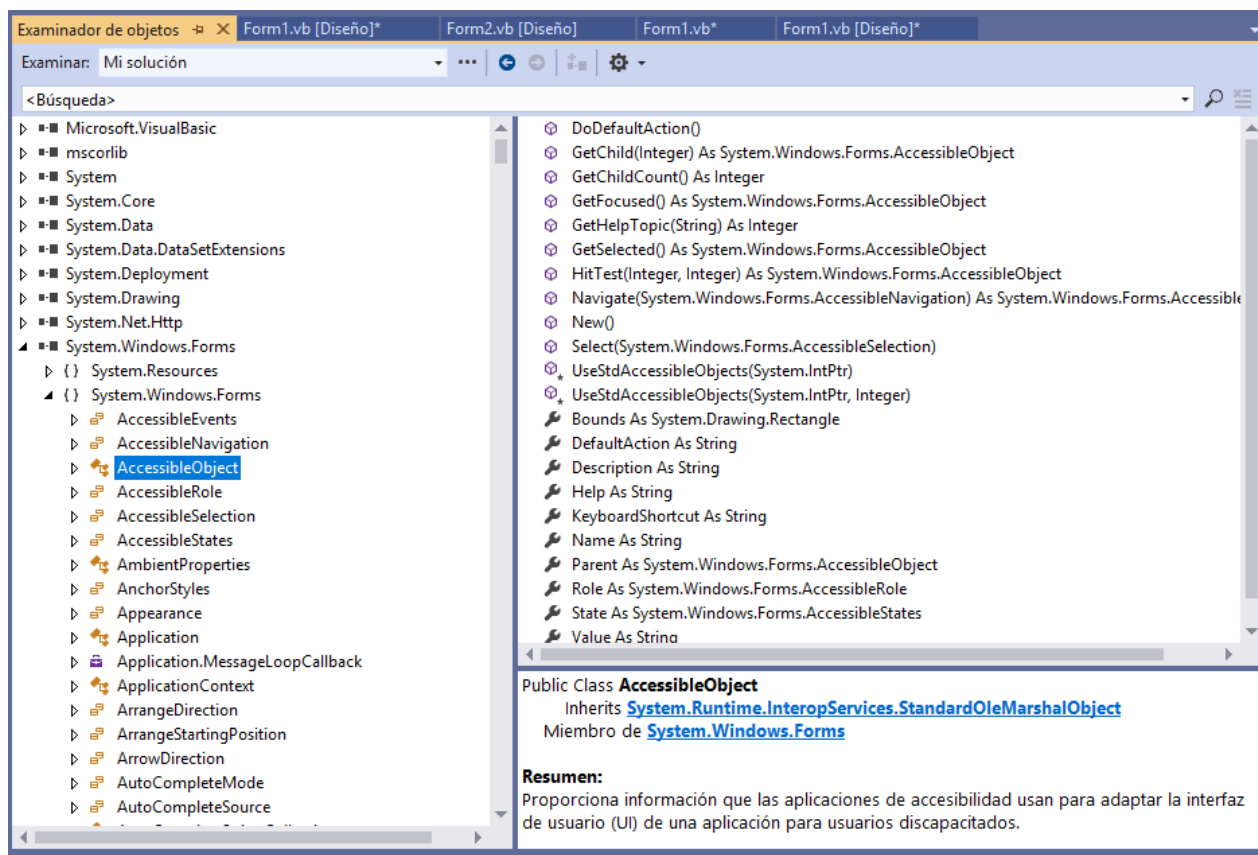
Está organizada en forma jerárquica, esto es, un nivel con el objeto o elemento más importante y con él un símbolo "▶" para desplegar el siguiente nivel, luego si tienen más subniveles podrás ir navegando por todos los elementos de la clase.

En el ejemplo de la derecha fijaros que está el nombre de nuestro programa y después dos formularios, que es justo los elementos que tiene el programa. Si expandimos el formulario vemos que están los procedimientos que aparecen ocultos en una "región" en el código y si expandimos el Form1 aparecen todas las propiedades, métodos y eventos de este objeto (formulario).



## 1.15. Explorador de objetos

Esta ventana nos será en muchas ocasiones bastante útil para poder explorar las capacidades de cada objeto. Para mostrarla utilizaremos el menú *Ver* y luego en "Examinador de objetos":



Habrás podido comprobar que se ha acoplado en la ventana principal y se ha añadido como una solapa más.

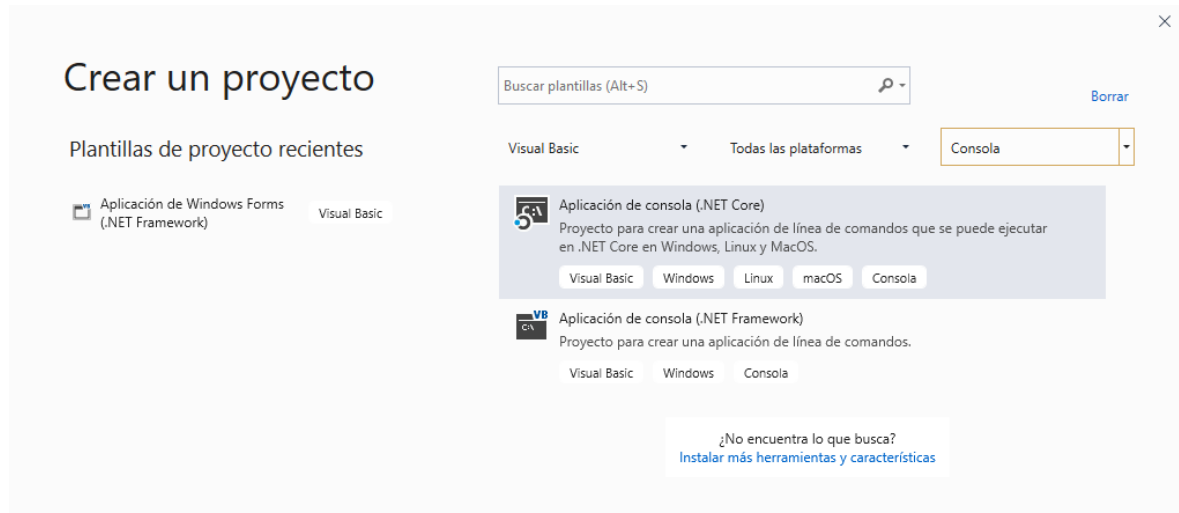
Así como en el caso anterior podíamos navegar a través de los objetos de nuestro programa, en esta pantalla podemos navegar por todos los objetos de VS.NET. Otra vez aparece el famoso objeto (o mejor dicho espacio de nombres o spacenames) llamado "System.Windows.Forms". En el panel de la derecha muestra la clase o elemento seleccionado y en la parte inferior la declaración.

## 1.16. La ayuda

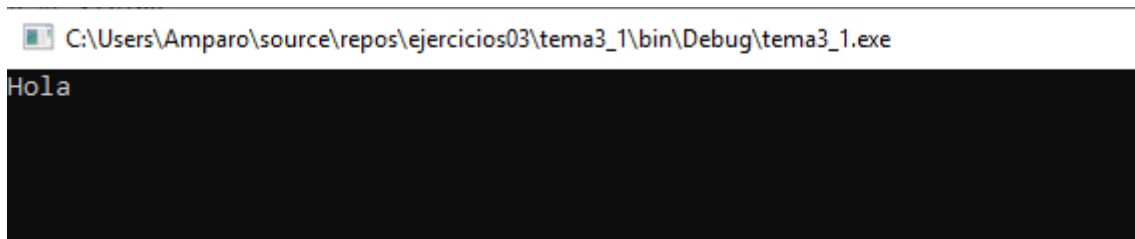
Y por fin otro de los elementos más utilizados en el IDE: la ayuda. Si hay una cosa en que Microsoft tenga especial cuidado es en la ayuda y su traducción. En este caso ha hecho un trabajo magnífico en cuanto a volumen y adaptación a nuestro idioma. Tenemos muchas formas de obtener ayuda.

## 2. Aplicaciones de consola

Con nuestro Visual Basic .NET podemos realizar muchos proyectos diferentes. Uno de ellos es la aplicación de consola:



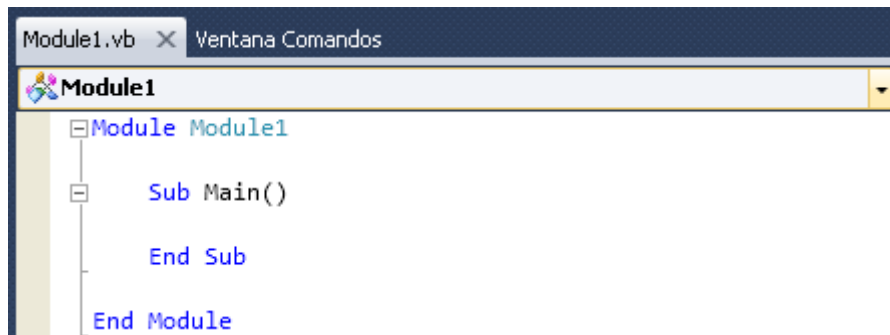
Las aplicaciones de consola son aquellas que se ejecutan en una ventana de comandos, como si fuera una aplicación de Ms-DOS:



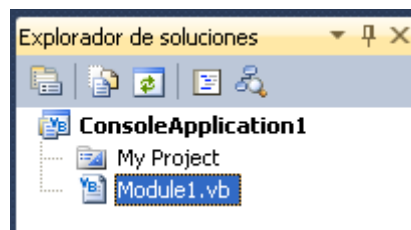
Nos serán muy útiles cuando necesitemos realizar pruebas en las que no necesitemos formularios ya que se ejecutarán mucho más rápido. También pueden ser útiles para crear programas de consola completos que realicen una serie de acciones: leer ficheros de texto y procesarlos, ...

### 2.1. Crear una aplicación de consola

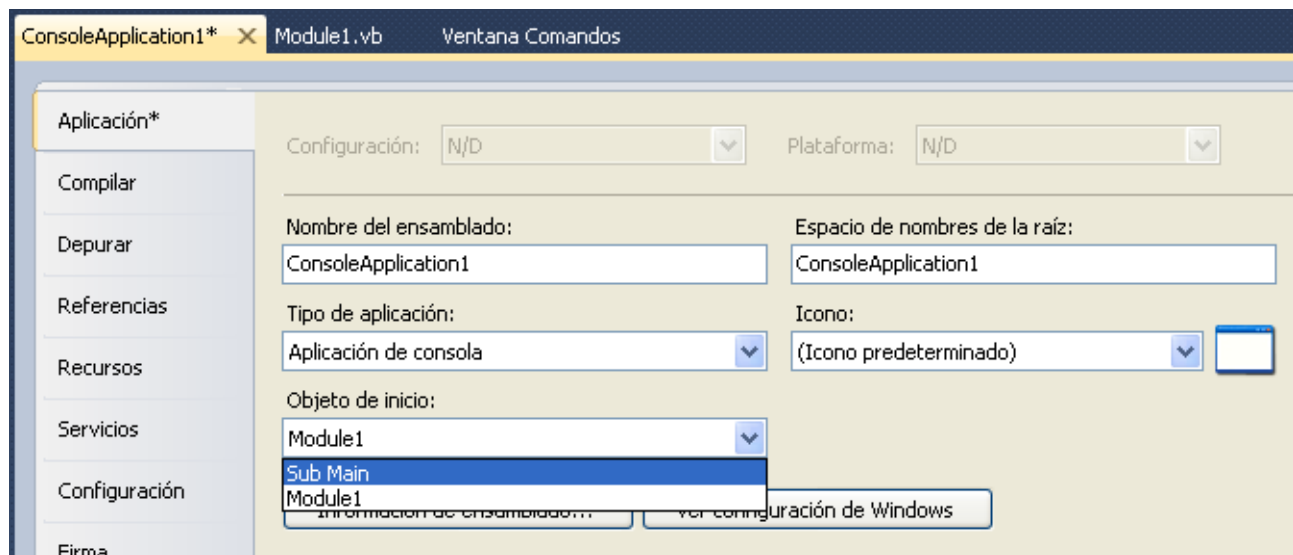
Como hemos visto en la anterior pantalla, crearemos un nuevo proyecto y le decimos que sea "Aplicación de consola". Veamos el código que nos muestra una vez que está preparado el IDE:



No tendremos la complejidad de los otros programas. Basta con ver en el explorador de soluciones que el proyecto se compone de un fichero Module1.vb:



Y que dentro del código tenemos el módulo dentro de las instrucciones "**Module** y **End Module**". Las instrucciones del procedimiento principal, donde va a comenzar nuestro programa, aparecen por defecto en **Sub Main () ... End Sub**. Si añadimos más módulos podemos indicarle cuál es el principal o punto de entrada al programa con la pantalla de las propiedades del programa consola1:



## 2.2. La clase Console

Como ya sabemos todo en VB.NET son clases. Esta clase se encuentra dentro del espacio de nombres System.

Recuerda que System designa al espacio de nombres principal o raíz, a partir del cual, descienden todos los espacios de nombres y clases de la plataforma. Otros espacios de nombres que veremos durante el curso serán: System.IO para los ficheros, System.Drawing para dibujos y System.Windows.Forms para las aplicaciones Windows.

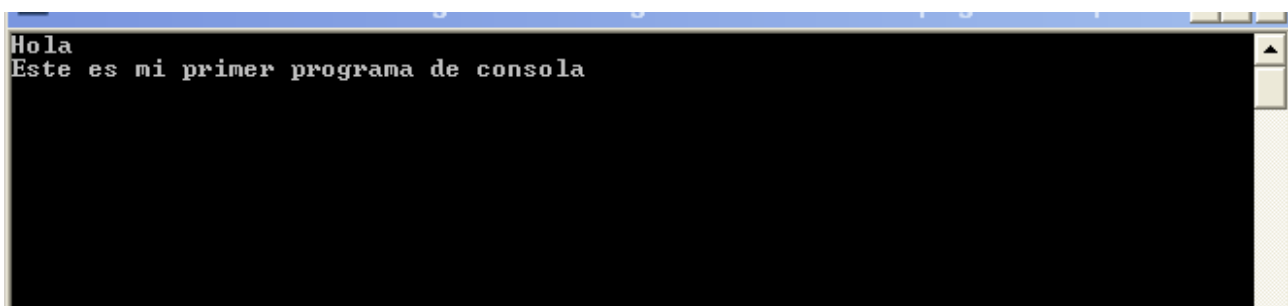
## 2.3. Escribir resultados

Para escribir textos en pantalla utilizaremos el método **WriteLine()** del objeto Console. Este método escribe en la línea actual el valor que le pasemos por parámetro añadiendo al final un "intro" o final de línea lo que provocará que el cursor pase a la siguiente línea.

Como primera aplicación escribiremos:

```
Module Module1
    Sub Main()
        Console.WriteLine("Hola")
        Console.WriteLine("Este es mi primer programa de consola")
    End Sub
End Module
```

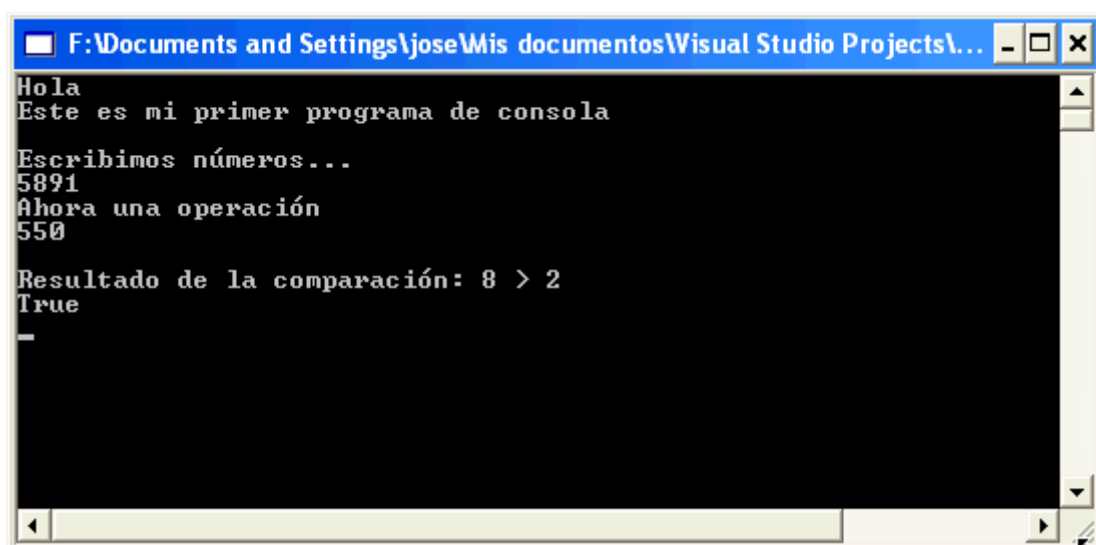
Ejecuta la aplicación y verás que funciona correctamente, aunque no llegamos a ver nada. Vamos a mejorarla poniendo algo parecido a una pausa al final. Podemos poner debajo de las dos instrucciones "**Console.ReadLine()**", esta instrucción lee datos por teclado así que esperará hasta que pulsemos Intro, de esta forma se nos quedará visible en pantalla:



Una variante de WriteLine es Write cuya única diferencia es que no avanza la línea al ejecutarse. Veamos ahora un sencillo programa donde se escribe textos de distintos tipos incluyendo una línea en blanco como separadora:

```
Sub Main()  
    'Escribimos dos líneas  
    Console.WriteLine("Hola")  
    Console.WriteLine("Este es mi primer programa de consola")  
  
    'Escribimos una línea vacía:  
    Console.WriteLine()  
  
    'Ahora escribimos números:  
    Console.WriteLine("Escribimos números...")  
    Console.WriteLine(5891)  
  
    'Ahora una operación  
    Console.WriteLine("Ahora una operación")  
    Console.WriteLine(200 + 350)  
  
    'Escribimos otra línea vacía:  
    Console.WriteLine()  
  
    'Escribimos una comparación lógica  
    Console.WriteLine("Resultado de la comparación: 8 > 2")  
    Console.WriteLine(5 > 2)|  
  
    Console.ReadLine()  
End Sub
```

Que nos producirá el resultado:



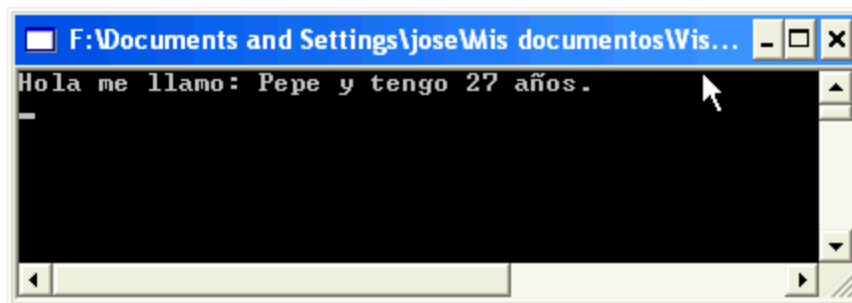


## 2.4. Escribir varios valores en una sola línea

Si queremos concatenar varios textos en una línea los podemos concatenar con el símbolo "&", tanto para variables como para textos. Por ejemplo:

```
'Tenemos dos variables que primero hay que declarar
Nombre="Pepe"
Numero=27
Console.WriteLine ("Hola me llamo: " & nombre & " y tengo " &
numero & " años.")
console.ReadLine
```

Así que irá concatenando la cadena con "Hola me llamo:" con nombre que es una variable, así que el resultado final será:



Ahora nos queda ver cómo podemos poner parámetros sustituibles.

Hemos visto como con **WriteLine** podemos indicar, entre otras cosas, una cadena que se mostrará en la ventana de consola, dentro de esa cadena podemos usar unos indicadores o marcadores en los que se mostrarán los parámetros indicados en esos dos métodos de la clase Console.

Los marcadores se indican con un número encerrado entre llaves, los números indicarán el número u orden del parámetro que siga a la cadena en el que está ese marcador. El primer parámetro estará representado por el cero y así sucesivamente.

Por ejemplo, para mostrar los valores de dos parámetros usaremos algo como esto:

**Console.WriteLine("{0} {1}", i, j)**

En este ejemplo, el {0} se sustituirá por el valor de la variable **i**, y el {1} por el valor de la variable **j**.

Que es mucho más cómodo que ir concatenando cadenas de caracteres y variables... fíjate en la diferencia:

```
Console.WriteLine ("Hola me llamo: " & nombre & " y tengo " & numero & " años.")  
Console.WriteLine ("Hola me llamo: {0} y tengo {1} años.", nombre, numero)
```

El segundo caso dice que se van a escribir dos variables en las posiciones de {0} y {1}. Y esas variables están a continuación en "nombre, numero" Cuando se indican marcadores, estos deben coincidir con el número de parámetros, es decir, si usamos el marcador {3}, debería existir ese parámetro, en caso de que haya menos de 4 parámetros (de 0 a 3), se producirá una excepción (un error).

**Nota:** Los parámetros pueden ser numéricos, de cadena, e incluso objetos basados en una clase. En todos los casos se usará la "versión" de tipo cadena de esos parámetros,

Con uno de estos marcadores, además de indicar el número de parámetro que se usará, podemos indicar la forma en que se mostrará: cuántos caracteres se mostrarán y si se formatearán a la derecha o la izquierda; también se pueden indicar otros valores de formato.

La forma de usar los marcadores o las especificaciones de formato será la siguiente:

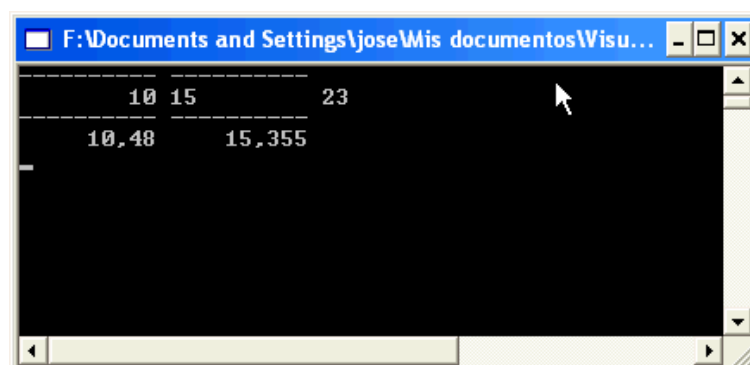
{ N [, M ][: Formato ]}

- **N** será el número del parámetro, empezando por cero.
- **M** será el ancho usado para mostrar el parámetro, el cual se rellenará con espacios. Si M es negativo, se justificará a la izquierda, y si es positivo, se justificará a la derecha.
- **Formato** será una cadena que indicará un formato extra a usar con ese parámetro.

Veamos algunos ejemplos para aclarar todo esto:

```
Console.WriteLine("-----")  
Console.WriteLine("{0,10} {1,-10} {2}", 10, 15, 23)  
'Console.WriteLine("-----")  
Console.WriteLine("{0,10:##.##} {1,10}", 10.476, 15.355)
```

El resultado de este ejemplo sería este:



En el primer caso, el primer número se muestra con 10 posiciones, justificado a la derecha, el segundo se justifica a la izquierda.

En el segundo caso, el número 10,476 se justifica también con 10 posiciones a la derecha, pero el número se muestra con un formato numérico en el que sólo se muestran dos decimales, por eso se ha redondeado el resultado. Sin embargo, en el segundo valor se muestran todos los decimales, ya que no se ha especificado ningún formato especial.

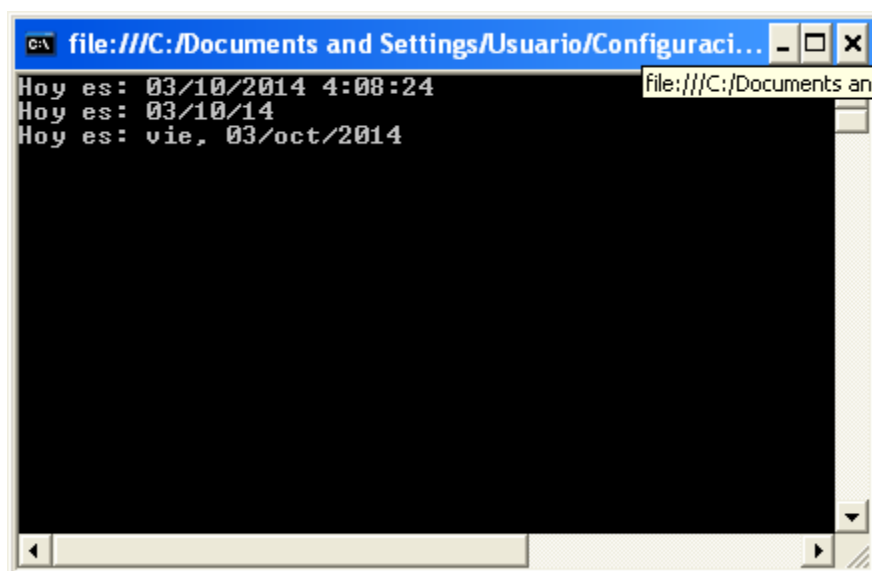
Estas especificaciones de formato son aplicables a números, pero también lo es a cadenas de caracteres y fechas. Veamos algunos ejemplos más:

```
Console.WriteLine("Hoy es: {0:G}", DateTime.Now)
```

```
Console.WriteLine("Hoy es: {0:dd/MM/yy}", DateTime.Now)
```

```
Console.WriteLine("Hoy es: {0:ddd, dd/MMM/yyyy}", DateTime.Now)
```

El resultado de estas líneas será el siguiente:

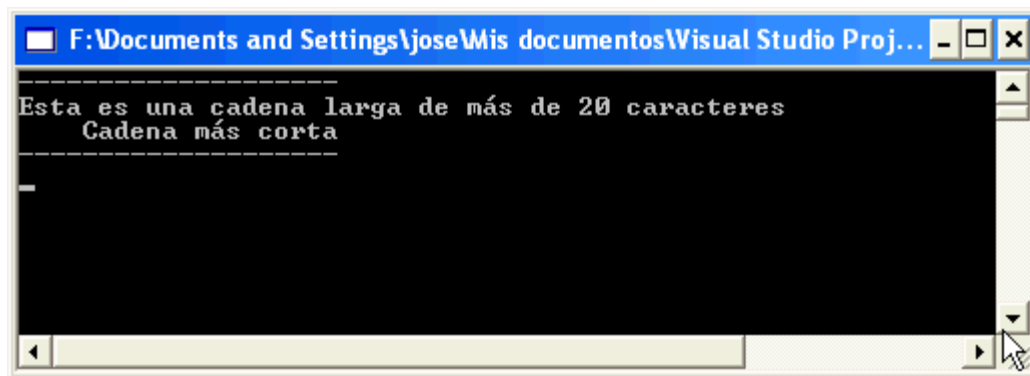


Como puedes comprobar, en esta ocasión no se indica cuantos caracteres se usan para mostrar los datos, (ya que sólo se indica el parámetro y el formato a usar), pero en el caso de que se especifiquen, si la cantidad de caracteres a mostrar es mayor que los indicados, no se truncará la salida, es decir, no se quitarán los caracteres que sobren. Esto lo podemos ver en el siguiente ejemplo:

```
Dim s As String = "Esta es una cadena larga de más de 20 caracteres"  
Dim s1 As String = "Cadena más corta"
```

```
Console.WriteLine("-----")  
Console.WriteLine("{0,20}", s)  
Console.WriteLine("{0,20}", s1)  
Console.WriteLine("-----")
```

La salida de este código es la siguiente:



Tal como podemos comprobar, la primera cadena excede los 20 caracteres (los indicados con las rayas), por tanto, se muestra todo el contenido, sin embargo la segunda cadena mostrada sí se posiciona correctamente con espacios a la izquierda hasta completar los 20 especificados.

En formato se pueden usar muchos caracteres diferentes según el tipo de datos a "formatear", la lista sería bastante larga y de momento nos vale con lo que hemos visto aquí.

## 2.5. Leer valores

Para leer valores por teclado utilizaremos como hemos visto antes el método `ReadLine`. Si queremos asignar el valor leído a una variable lo indicaremos en la asignación a la variable:

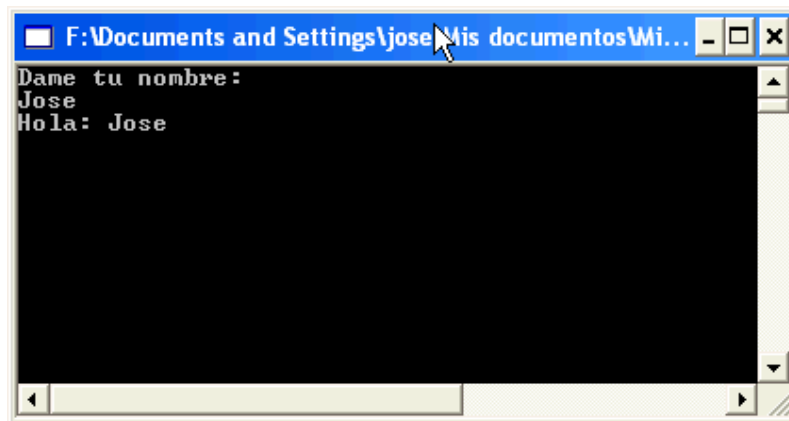
'Declaro una variable de tipo cadena de caracteres

```
Dim cadena as String  
cadena=Console.ReadLine()
```

## Ejercicios

### Ejercicio 1

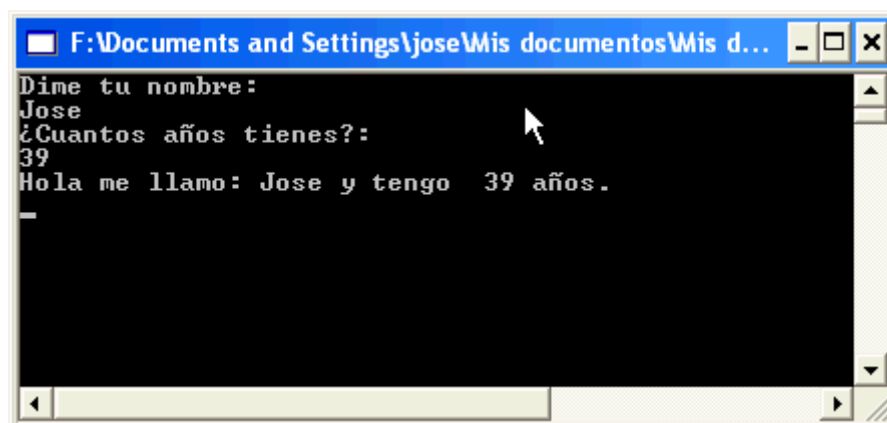
Haz un sencillo programa que lea un dato por pantalla y lo escriba:



```
F:\Documents and Settings\jose\Mis documentos\Mis d...
Dame tu nombre:
Jose
Hola: Jose
```

### Ejercicio 2

Crea otro programa de consola que lea el nombre y la edad y los escriba en pantalla. Debe utilizar la notación de escritura de variables con {}:



```
F:\Documents and Settings\jose\Mis documentos\Mis d...
Dime tu nombre:
Jose
¿Cuántos años tienes?:
39
Hola me llamo: Jose y tengo 39 años.
```

### Ejercicio 3

Practica con el IDE creando un formulario con varios controles. Luego localiza las ventanas de "Explorador de Soluciones", "Propiedades", "Pantalla de código", "Cuadro de herramientas" y Vista de clases.