

# 01 – PROGRAMACIÓN I

---

MG. NICOLÁS BATTAGLIA



**UAIOnline**  
*ultra* >>>

# UNIDAD I

---

INTRODUCCIÓN A LA PROGRAMACIÓN VISUAL - CLASE I  
REPASO E INTRODUCCIÓN A LA MATERIA



**UAIOnline**  
*ultra* >>>

# OBJETIVOS

- Incorporar los conocimientos para identificar las diferencias entre las metodologías de programación estructurada, la orientada a objetos y orientadas a eventos.
- Dominar el manejo de punteros y archivos desde ambas metodologías
- Incorporar los elementos brindados por la interfaz de los lenguajes orientados a eventos con el objeto de poder aplicarlos en la construcción de software.
- Dominar los aspectos lógicos y algorítmicos de la programación orientada a eventos con el objeto de poder aplicarlos en la construcción de software.
- Desarrollar la idea fundamental de objeto, las propiedades que la definen y los eventos que lo controlan.
- Comprender las técnicas de acceso a archivos, su administración y las ventajas y las desventajas que cada una representa.

# APROBACIÓN DE LA MATERIA

- 2 Parciales
- Recuperatorios
- TP de cada unidad
- Asistencia
  - 75% de presentismo

# BIBLIOGRAFÍA OBLIGATORIA

- Harvey Deitel-Paul Deitel, “C# como programar”, Mexico, Pearson Prentice Hall, 2007
- Brizuela, Rafael , “Apuntes de programacion I” , Buenos Aires, UAI , 2016.
- Ceballos, Fco Javier, “Microsoft C# lenguaje y aplicaciones”, Mexico, Alfaomega RAMA 2008
- Nilsson, Nils J. Inteligencia artificial: una nueva síntesis.-- Madrid: McGraw-Hill, c2001

---

REPASO GENERAL

**UAIOnline**  
*ultra* >>>

# TEMAS

- Variables
- Vectores
- Matrices
- Funciones y procedimientos
- Estructuras y diagramación lógica.

# VARIABLES

- Las variables almacenan valores que pueden cambiar cuando una aplicación se está ejecutando
- Las variables tienen seis elementos básicos:

Elemento	Descripción
Nombre	La palabra que identifica la variable en código
Dirección	La ubicación de memoria donde se almacena el valor
Tipo de datos	El tipo y tamaño inicial de datos que la variable puede almacenar
Valor	El valor en la dirección de la variable
Ámbito	El conjunto de todo el código que puede acceder y utilizar la variable
Vida	El intervalo de tiempo durante el cual una variable es válida



# VARIABLES

- Reglas para poner nombres
  - Empezar con un carácter alfabético o guión bajo
  - No utilizar espacios ni símbolos
  - No utilizar palabras clave como **int**
- Ejemplos de nombres de variables
  - NombreC1iente (PascalCasing)
  - numeroCuenta (camelCasing)

- Sintaxis para declarar variables
  - *Tipo de dato* *NombreVariable* ;
- Ejemplos de variables de tipo valor

```
int Divisor;  
double Dividendo;  
char x,y,z;
```

- Ejemplos de variables de tipo referencia

```
form Formulario;  
string Apellido;
```

- Sintaxis para declarar variables
  - *Tipo de dato* *NombreVariable* ;
- Ejemplos de variables de tipo valor

```
int Divisor;  
double Dividendo;  
char x,y,z;
```

- Ejemplos de variables de tipo referencia

```
form Formulario;  
string Apellido;
```

- Sintaxis para asignar valor

```
int Numero = 1517;
```

Tipo de dato

Nombre variable

Valor asignado

# OPEADORES LOGICOS

Categoría	Expresión	Descripción
AND lógico	$x \& y$	AND bit a bit entero, AND lógico booleano
XOR lógico	$x \wedge y$	XOR bit a bit entero, XOR lógico booleano
OR lógico	$x \mid y$	OR bit a bit entero, OR lógico booleano
AND condicional	$x \&\& y$	Evalúa y solo si x es true
OR condicional	$x \mid\mid y$	Evalúa y solo si x es false
Uso combinado de NULL	$x \ ?? \ y$	Se evalúa como y si x es NULL; de lo contrario, se evalúa como x
Condicional	$x \ ? \ y : z$	Se evalúa como y si x es true y como z si x es false

# VECTORES Y MATRICES

- **Estaticos**

- `Int[] array = new int[2]`

- **Dinámicos**

- `Array.Resize(ref array,4);`

- **Matiz** (Array multidimensional)

- `Int[,] matriz = new int[100,200]`

- Un elemento es un valor en un ARRAY, y su longitud es el numero total de elementos que puede contener.
- Posee un limite inferior de CERO por default y cualquier numero de limite superior
- Tiene capacidad fija
- Para ampliar capacidad hay que crear nuevo array o usar RESIZE (copia de forma transparente)
- Tamaño máximo: 2 gb en 64 bits

```
char[] array = new char[4];
```

```
Array.Resize(ref array, 2);
```

# ESTRUCTURAS DE REPETICIÓN

- En C# las entradas y salidas se manejan como flujos
- **Salida de datos por Pantalla**
  - `Console.WriteLine("Introduzca un texto");`  
`String texto;`
- **Entrada por teclado**
  - `texto=Console.ReadLine();`
  - `Console.WriteLine("El texto introducido es: " + texto);`



---

# INTRODUCCIÓN A .NET C#.

WindowsFormsApp1

Archivo Editar Ver Proyecto Compilar Depurar Formato Prueba Analizar Herramientas Extensiones Ventana Ayuda Buscar (Ctrl+Q)

Debug Any CPU Iniciar

Form1.cs\* Form1.cs [Diseño]\*

Orígenes de datos

Búsqueda en el Cuadro de herramie

MessageQueue  
MonthCalendar  
NotifyIcon  
NumericUpDown  
OpenFileDialog  
PageSetupDialog  
Panel  
PerformanceCounter  
PictureBox  
PrintDialog  
PrintDocument  
PrintPreviewControl  
PrintPreviewDialog  
Process  
ProgressBar  
PropertyGrid  
RadioButton  
RichTextBox  
SaveFileDialog  
SerialPort  
ServiceController  
SplitContainer  
Splitter  
StatusStrip  
TabControl  
TableLayoutPanel  
TextBox  
Timer  
ToolStrip  
ToolStripContainer  
ToolTip

Form1

Grupo 1

Botón 1

¿Hace frío?

Nombre

Explorador de proyectos

Cuadro de herramientas

Cuadro propiedades y eventos

Explorador de soluciones

Búsqueda en Explorador de soluciones (Ctrl+)

Solución "WindowsFormsApp1" (1 de 1 proyecto)

WindowsFormsApp1

Properties  
Referencias  
App.config  
Form1.cs  
Form1.Designer.cs  
Form1.resx  
Program.cs

Propiedades

checkBox1 System.Windows.Forms.CheckBox

(DataBindings)

AppearanceChanged  
AutoSizeChanged  
BackColorChanged  
BackgroundImageChanged  
BackgroundImageLayoutCh  
BindingContextChanged  
CausesValidationChanged  
ChangeUICues  
CheckedChanged checkBox1\_CheckedChanged  
CheckStateChanged  
Click checkBox1\_Click  
ClientSizeChanged  
ControlNameChanged

CheckedChanged

Tiene lugar cuando cambia la propiedad Checked.

# NOMBRES DE ESPACIOS

- Agrupamientos lógicos de funciones dentro de librerías en .NET
- El nombre de espacios del framework de .net comienza con “System”.
- Al crear programas en .NET se define un nombre de espacios para todas sus funciones.
- En "System.Console" esa palabra "System" indica que las funciones que estamos usando pertenecen a la estructura básica de C# y de la plataforma .Net.
- “using System.Linq” por ejemplo, permite importar las librerías dentro de ese nombre de espacios.

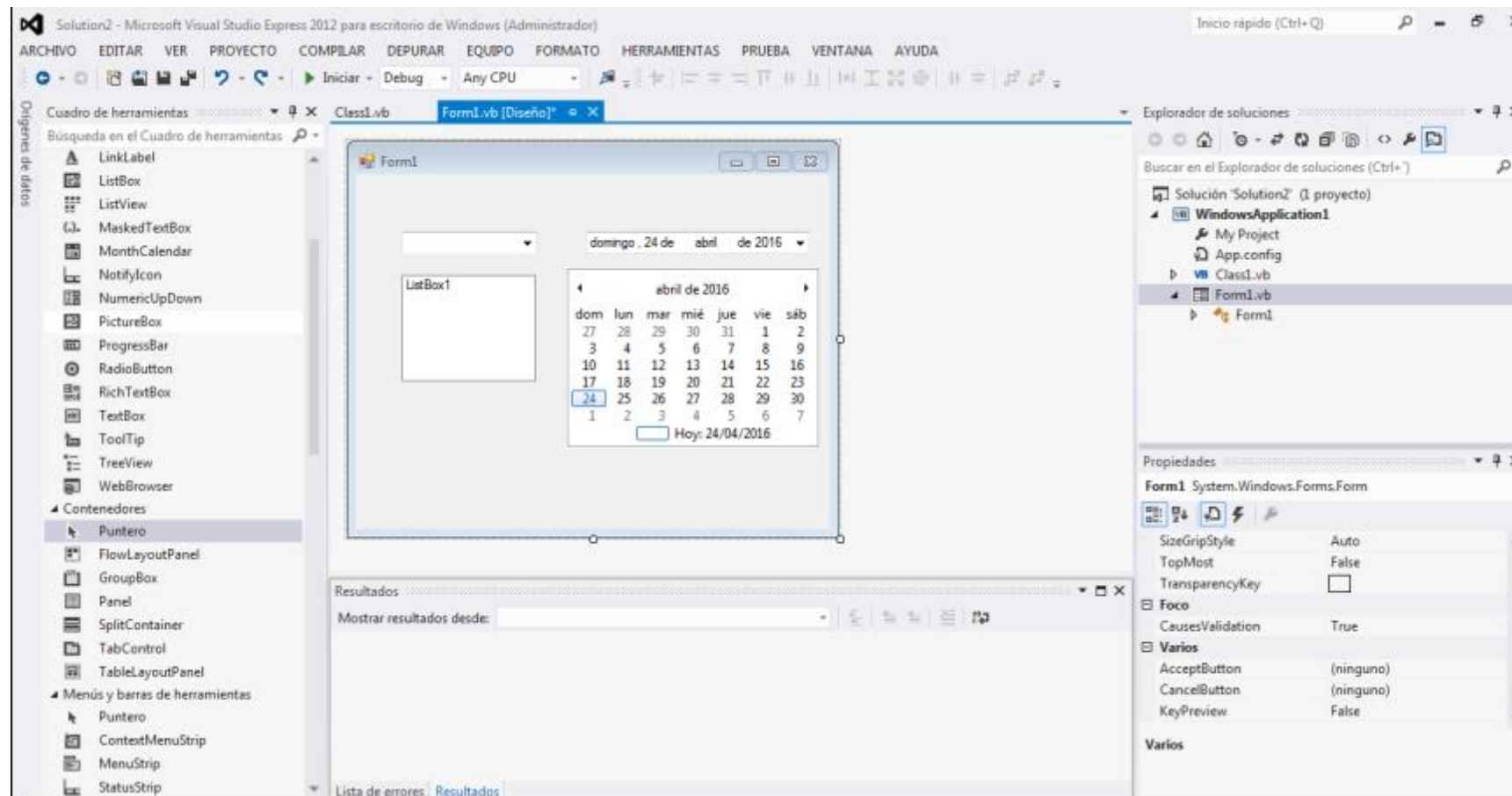
# ENTRADAS Y SALIDAS EN C#

- En C# las entradas y salidas se manejan como flujos
- **Salida de datos por Pantalla**
  - `Console.WriteLine("Introduzca un texto");`  
`String texto;`
- **Entrada por teclado**
  - `texto=Console.ReadLine();`
  - `Console.WriteLine("El texto introducido es: " + texto);`

# VISUAL STUDIO - C#

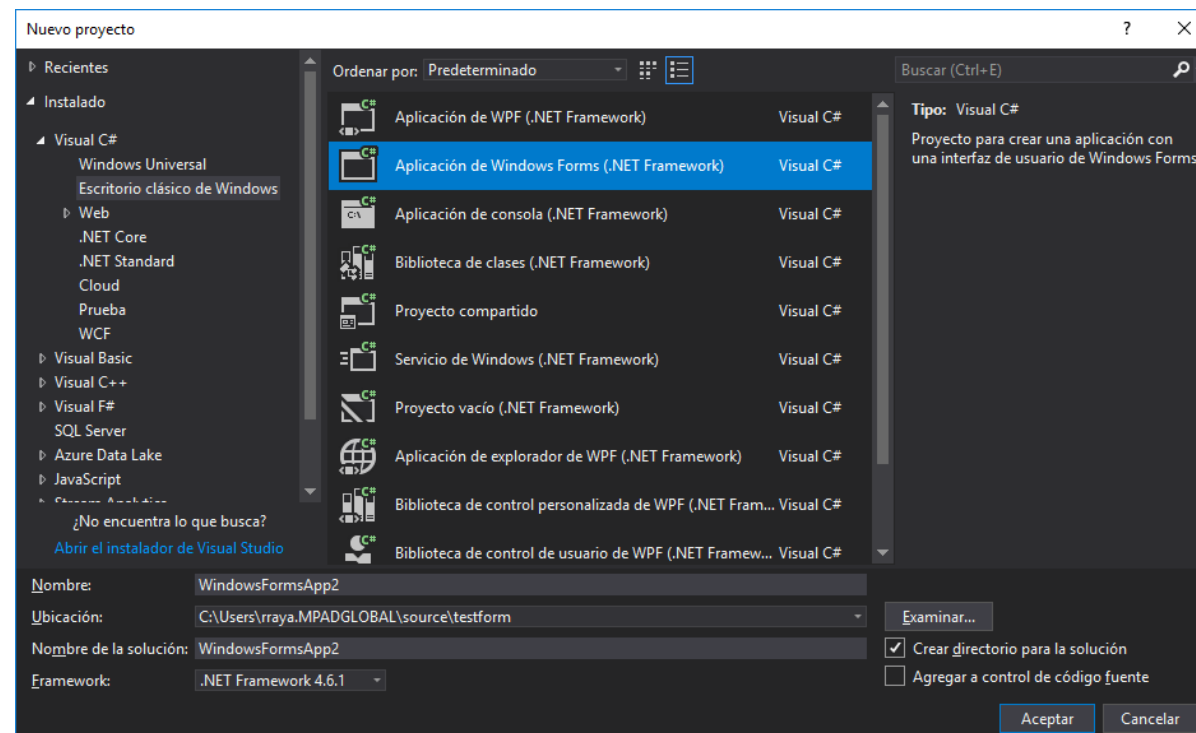
- VS.NET simplifica el desarrollo de aplicaciones basadas en .NET proporcionando un entorno de desarrollo simple y unificado
- Características
  - Un solo IDE (Integrated Development Environment)
  - Soporte para varios lenguajes .NET (VB.NET, C#,...)
  - Desarrollo de múltiples tipos de proyectos
  - Explorador Web integrado (basado en IE)
  - Interface personalizable
  - Posee varias utilidades adicionales: Acceso a datos SQL Server, Depurador, Intellisense, Emuladores para móviles, etc.

# VISUAL STUDIO - C#



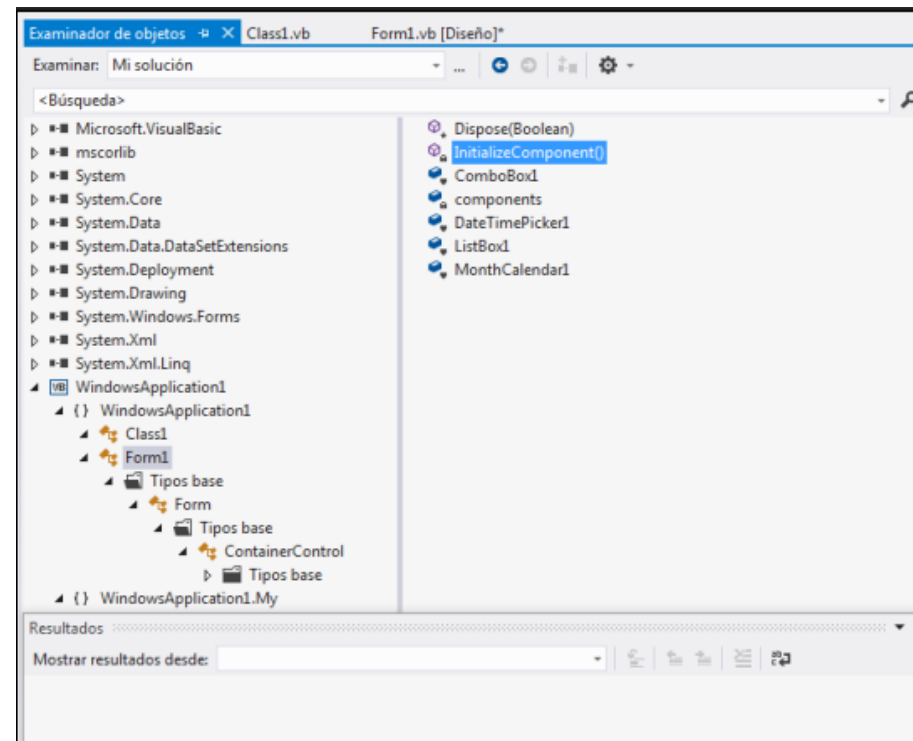
# VISUAL STUDIO - C#

- VS.NET permite crear varios tipos de proyectos
- VS.NET crea todos los archivos necesarios automáticamente



# VISUAL STUDIO - C#

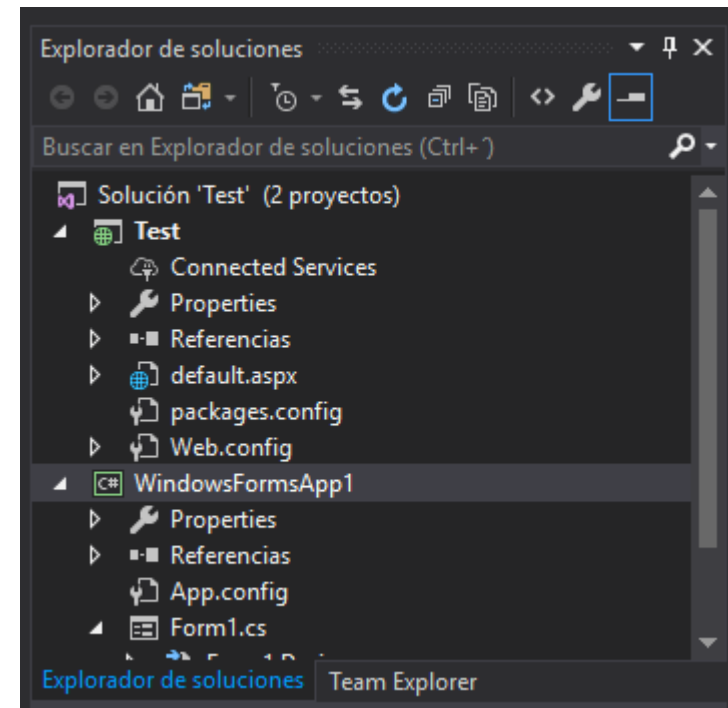
- Examinador de objetos
  - Proporciona información detallada sobre los objetos, sus propiedades, métodos, eventos, constantes, etc.





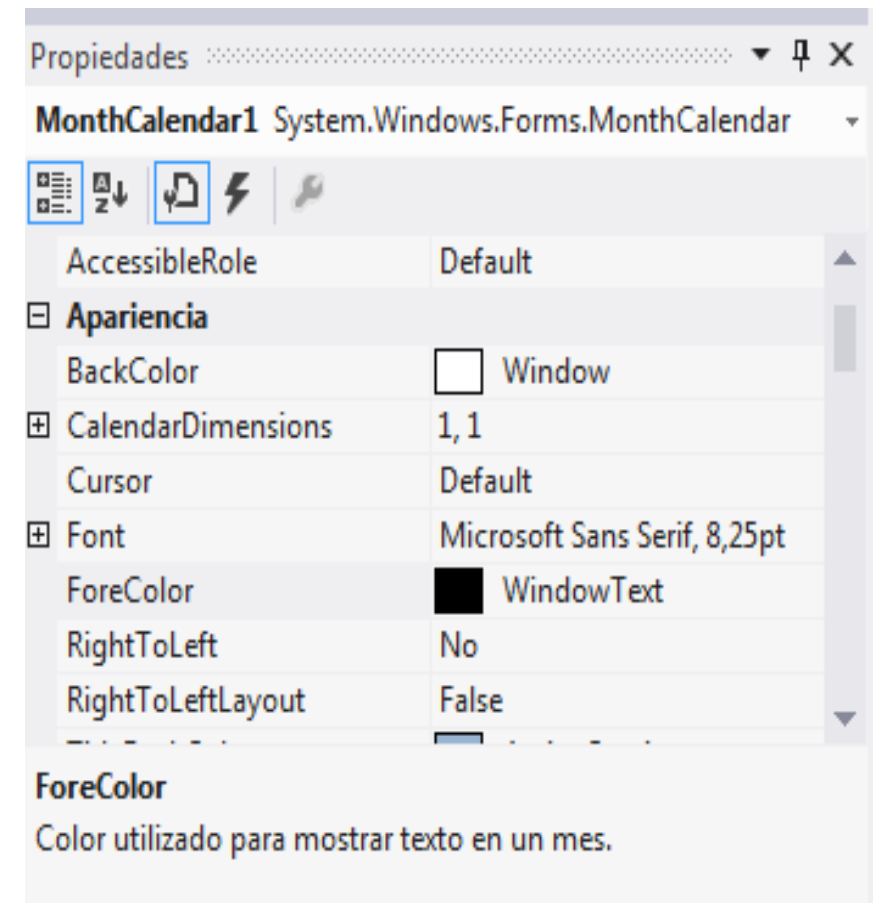
# VISUAL STUDIO - C#

- Explorador de soluciones
  - Muestra los archivos de/los proyectos de la solución
  - Permite eliminar y mover los archivos del proyecto
  - Permite agregar nuevos elementos al proyecto
  - Establecer referencias a assemblies y servicios web
  - Crear carpetas
  - Etc.

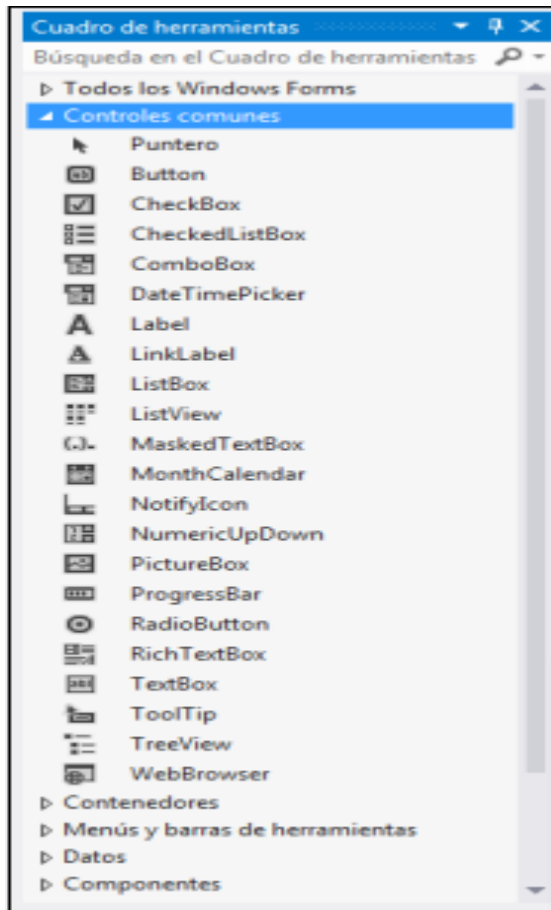


# VISUAL STUDIO - C#

- Ventana de Propiedades
- Permite acceder y modificar a las propiedades y eventos del objeto seleccionado (WebForm, control, clase, etc.)



# VISUAL STUDIO - C#



## ■ Caja de herramientas (Toolbox)

- Contiene los componentes y controles que vamos a utilizar en nuestros WebForms

# VISUAL STUDIO - C#

- Modo depuración (debugger)
  - Detiene la operación de una aplicación
  - En modo de interrupción, podemos:
    - Recorrer nuestro código línea por línea
    - Determinar los procedimientos activos que se han invocado
    - Observar los valores de variables, propiedades y expresiones
    - Utilizar las ventanas de depuración para modificar valores de variables y propiedades
  - Cambiar el flujo del programa
  - Ejecutar instrucciones de código

# VISUAL STUDIO - C#

- Puntos de interrupción
  - Un punto de interrupción es un marcador en nuestro código que hace que Visual Basic detenga la ejecución del código en una línea específica
  - No podemos colocar puntos de interrupción en código no ejecutable



# VISUAL STUDIO - C#

- Recorrer el código
  - Paso a paso por instrucciones o por procedimientos: ejecuta la siguiente línea de código; si la línea siguiente contiene una invocación a un procedimiento:
    - **Paso a paso por instrucciones:** únicamente ejecuta la invocación, y se detiene en la primera línea de código dentro del procedimiento
    - **Paso a paso por procedimientos:** ejecuta todo el procedimiento, y se detiene en la primera línea de código fuera del procedimiento

# ENTRADAS Y SALIDAS EN C#

- En C# las entradas y salidas se manejan como flujos
- **Salida de datos por Pantalla**
  - `Console.WriteLine("Introduzca un texto");`  
`String texto;`
- **Entrada por teclado**
  - `texto=Console.ReadLine();`
  - `Console.WriteLine("El texto introducido es: " + texto);`

# ENTRADAS Y SALIDAS EN C#

- while
- do { //... }while(<expresión>);
- while(<expresión>) { //... }
- while(!<expresión>) { //... }
- do { //... }while(!<expresión>);
- For
- for (Poslcial = 0; Poshasta <= 4; Incremteno++)