Fundamentos de sintaxis en algunas instrucciones de C#

Identificadores

Un identificador le da nombre único a un elemento en un programa (Variables, procedimientos, etc.).

- No puede contener operadores como + - * /
- Debe iniciar con letra o el subguión (_)
- Puede tener cualquier longitud
- Puede contener mayúsculas y minúsculas
- En un identificador sí se hace diferencia entre mayúsculas y minúsculas. De esta manera, suma es diferente de Suma
- No debe ser una palabra reservada.

Algunos tipos de datos

int	entero
double	coma flotante
char	Un caracter
string	Cadena de caracteres
bool	Booleano (Verdadero, Falso)
DateTime	Fecha/Hora

Declaración de Variables

```
string  Nombre;
int     HorasTrabajadas;
double  SueldoPorHora;
char    Otro;
```

También se pueden inicializar las variables al mismo tiempo que se declaran:

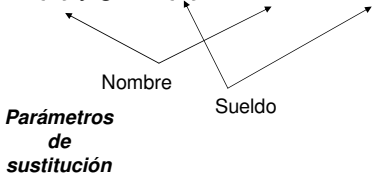
```
string Nombre = "Juan";
int    HorasTrabajadas = 40;
double SueldoPorHora = 150.00;
char   Otro = 'N';
```

Impresión (Proyectos de Consola)

System.Console.WriteLine (" Hola ");

System.Console.WriteLine(Sueldo);

System.Console.WriteLine
(" Mi nombre es **{0}** y gano **{1}** ", **Nombre**, **Sueldo**);



Impresión (Proyectos de consola)

```
System.Console.WriteLine (" Nombre: " + Nombre);
```

Concatenación

```
System.Console.WriteLine
(" Mi nombre es " + Nombre + " gano " + Sueldo);
```

Varias Concatenaciones

Caracteres de Escape

- Utilizados para representar caracteres no-imprimibles.

Ejemplos:

```
System.Console.WriteLine("Linea 1 \n Linea 2 ");
System.Console.WriteLine("Comillas: \"   ");
System.Console.WriteLine("Apostrofo \'  ");
System.Console.WriteLine("Barra \\ ");
System.Console.WriteLine("Tabulador \t Tabulador");
```

Caracteres de escape mas comunes

Alarma	\a
Barra	\\
Apostrofo	\’
Comillas	\”
Nueva Línea	\n
Tabulador	\t

Sentencias using

- Si se agregan al inicio de la clase, NO es necesario repetirlas en cada instrucción.
- Por ejemplo, si al inicio de la clase se agrega:

```
using System;
```

puede usarse durante toda la clase:

```
Console.WriteLine( " Hola Mundo " );
```

en lugar de:

```
System.Console.WriteLine( "Hola Mundo " );
```

Comentarios en el programa

```
// Esta es una linea de comentarios
// Esta es otra linea de comentarios

/* Estas son
varias lineas de
Comentarios */
```

NOTA: Los comentarios NO se ejecutan.
Solo sirven como documentación interna en el programa.

Pedir Datos (Proyectos de Consola)

```
Nombre = System.Console.ReadLine();
```

El resultado se almacena en esta variable

Pedir un dato String

```
HorasTrabajadas =
System.Convert.ToInt32(System.Console.ReadLine());
```

Conversión de String a Entero
TAMBIEN puede hacerse así:

```
HorasTrabajadas = int.Parse(System.Console.ReadLine());
```

Cálculos y Asignaciones

Si las variables ya están declaradas, los cálculos y asignaciones se pueden hacer directamente:

```
SueldoTotal = HorasTrabajadas * SueldoPorHora;
Descuento   = SueldoTotal * 0.08;
SueldoTotal = SueldoTotal – Descuento;
Bonos = 200;
SueldoTotal = SueldoTotal + Bonos;
```

Cálculos y Asignaciones

Si las variables NO están declaradas, se pueden declarar al momento de que aparecen solamente por primera vez:

```
double SueldoTotal = HorasTrabajadas * SueldoPorHora;
double Descuento   = SueldoTotal * 0.08;
       SueldoTotal = SueldoTotal – Descuento;
double Bonos = 200.00;
       SueldoTotal = SueldoTotal + Bonos;
```

Operadores Aritméticos

`+`	**Suma**
`-`	**Resta**
`*`	**Multiplicación**
`/`	**División**
`%`	**Residuo de división entera**
`=`	**Igualdad**
`++`	**Incremento en 1**
`--`	**Decremento en 1**

Jerarquía:

Primero se ejecutan `*` `/`
Después se ejecutan `+` `-`
Los paréntesis alteran la jerarquía

Operadores de asignación adicionales

- Sirven para simplificar expresiones de asignación. Ejemplos:
  - a += 5 → a = a + 5
  - a -= 5 → a = a - 5
  - a *= 5 → a = a * 5
  - a /= 5 → a = a / 5
- Ejemplos con los Operadores de Autoincremento:
  - x ++ → x = x + 1
  - x -- → x = x - 1

Uso de paréntesis en las Operaciones Aritméticas

Se pueden utilizar paréntesis en las Operaciones Aritméticas:

```
Resultado = (Num1 + Num2) * Num3;
```

Se pueden anidar paréntesis.
Los Paréntesis internos se ejecutan primero:

```
Resultado = Num1 + (Num2 * (Num3 + Num4) ) + Num5;
```

Operadores Relacionales

`>`	**Mayor que**
`<`	**Menor que**
`>=`	**Mayor o igual que**
`<=`	**Menor o igual que**
`==`	**Igual**
`!=`	**Diferente**

Utilizarlos en C#

Operadores Lógicos

`&&`	**AND**	**(Y)**
`||`	**OR**	**(O)**
`!`	**NOT**	**(NO)**

Utilizarlos en C#

Tablas de verdad

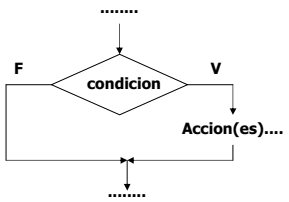
AND OR

Condicion1	Condicion2	Condicion1 && Condicion2	Condicion1 || Condicion2	! Condicion1 (NOT Condicion1)	! Condicion2 (NOT Condicion2)
V	V	V	V	F	F
F	V	F	V	V	F
V	F	F	V	F	V
F	F	F	F	V	V

Decisiones (if)

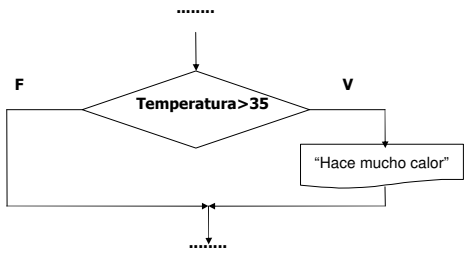
La Decisión puede ser SIN PARTE FALSA:

```
if (condicion)
{
  [Qué pasa si la condicion es Verdadera]
}
```



Decisiones (if) Ejemplo

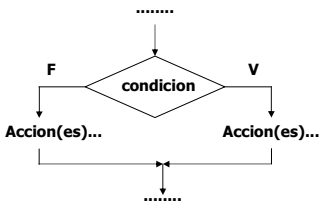
```
if (Temperatura > 35)
{
   System.Console.WriteLine("  Hace mucho calor " );
}
```



Decisiones (if – else)

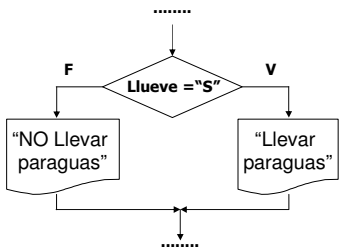
La Decisión puede incluir parte falsa:

```
if (condicion)
{
        [Qué pasa si la
        condicion es Verdadera]
}
else
{
        [Qué pasa si la
        condición es Falsa]
}
```



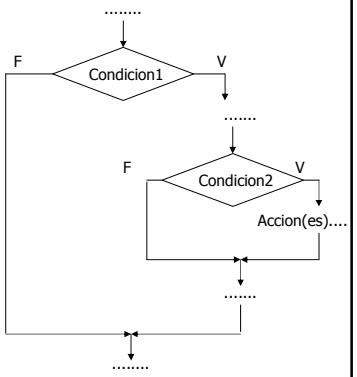
Decisiones (if – else) Ejemplo

```
if (Llueve == "S")
{
    System.Console.WriteLine
      ("Llevar paraguas");
}
else
{
    System.Console.WriteLine
     ("NO Llevar paraguas");
}
```



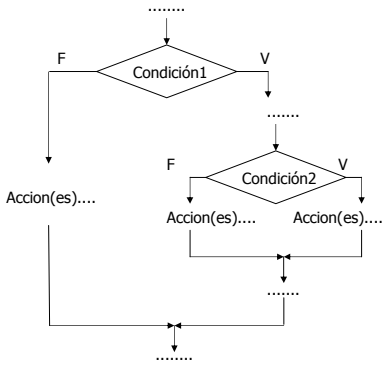
Decisiones Anidadas (if anidados)

```
if (condicion1)
{       [Qué pasa si se
        cumple la
        condicion1]
        if (condicion2)
        {       [Qué pasa si se
                cumple la
                condicion2]
        }
}
```



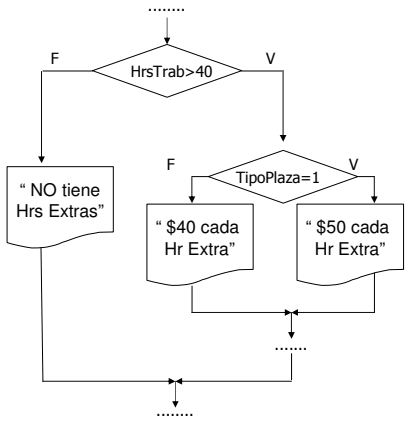
Decisiones Anidadas (if anidados)

```
if (condicion1)
{   [Qué pasa si se
    cumple la condicion1]
    if (condicion2)
    {       [Qué pasa si se
            cumple la condicion2]
    }
    else
    {       [Qué pasa si NO se
            cumple la condicion2]
    }
}
else
{       [Qué pasa si NO se
        cumple la condicion1]
}
```



Decisiones Anidadas (if anidados) Ejemplo

```
if (HrsTrab > 40)
{   if (TipoPlaza == 1)
    {     System.Console.WriteLine
          (" $50 pesos cada Hr Extra");
    }
    else
    {     System.Console.WriteLine
          (" $40 pesos cada Hr Extra");
    }
}
else
{     System.Console.WriteLine
      ("No tiene Horas Extras");
}
```



Selección Múltiple

```
int variable = 1;
switch  (variable)
{
    case 1:     System.Console.WriteLine("1");
                break;
    case 2:     System.Console.WriteLine("2");
                break;
    case 3:     System.Console.WriteLine("3");
                break;
    default:    System.Console.WriteLine("Otro");
                break;
}
```

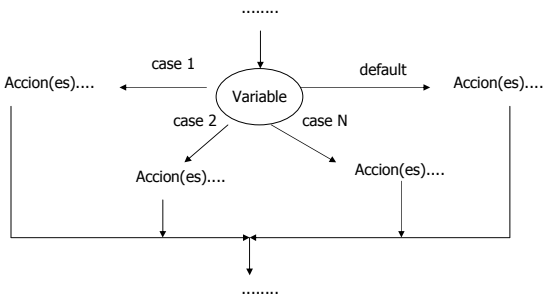
Variable que se desea checar

Posibles Casos (Valores que puede tener)

Si ningún caso anterior Se cumple…

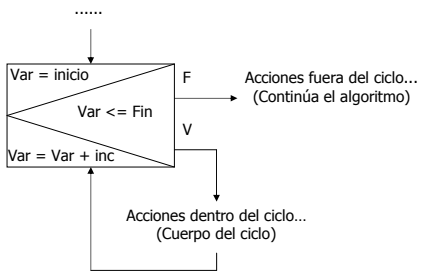
En cada "case" se pone un "break" Para que no continúe la ejecución con los demás casos hacia abajo.

Selección Múltiple



Ciclo for()

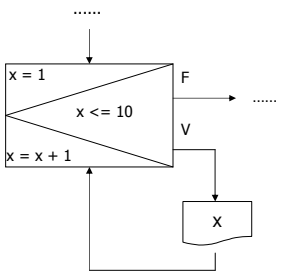
```
for (x=inicio; x<=fin ; x=x+incremento )
{

   ...cuerpo del ciclo...

}
```



Ciclo for() Ejemplo

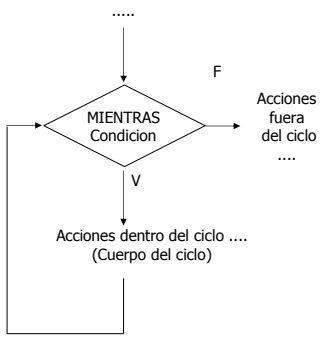
```
for (x=1; x<=10 ; x=x+1 )
{
   System.Console.WriteLine( x );
}
```

Este ciclo for imprime los numeros del 1 al 10



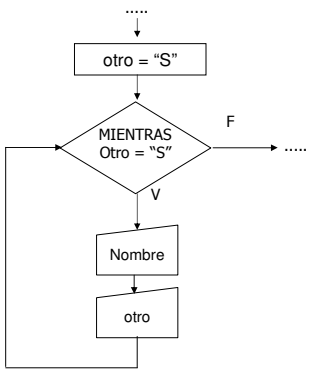
Ciclo while{ }

```
while ( condicion )
{
   ...cuerpo del ciclo...
}
```



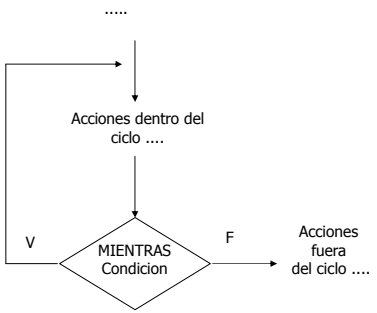
Ciclo while{ } Ejemplo

```
otro = "S";
while (otro == "S")
{
   System.Console.WriteLine
   (" Introduce el nombre: ");

   Nombre = System.Console.ReadLine();

   System.Console.WriteLine
   (" Hay mas? ");

   otro = System.Console.ReadLine();
}
```



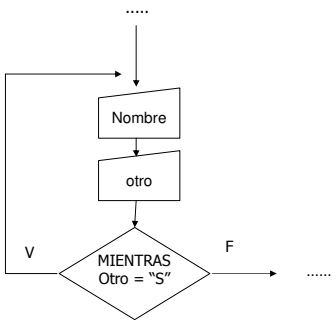
Ciclo do{ }while()

```
do
{
   ...cuerpo del ciclo...
} while ( condicion );
```



Ciclo do{ }while() Ejemplo

```
do
{
    System.Console.WriteLine
    (" Introduce el nombre: ");

    Nombre =
    System.Console.ReadLine();

    System.Console.WriteLine
    (" Hay mas? ");

    otro = System.Console.ReadLine();
} while ( otro =="S" );
```



Manejo de errores

```
try
{       [Bloque que puede causar errores]
}
catch
{       [Qué hacer si sucede un error]
}
finally
{       [De cualquier manera, hacer lo siguiente…]
}
```

Manejo de errores [Ejemplo]

```
static void Main(string[ ] args)
{
        try
        {  System.Console.WriteLine(" Introduce un numero: ");
           int a = System.Convert.ToInt32 (System.Console.ReadLine() );
        }
        catch
        {  System.Console.WriteLine(" Ha habido un error...");
        }
        finally
        {  System.Console.WriteLine(" Con error y Sin error, este mensaje aparece. ");
           System.Console.ReadLine();
        }
      }
```

Manejo de errores [Ejemplo – Parte 2]

```
static void Main(string[ ] args)
    {
      try
      {  System.Console.WriteLine(" Introduce un numero: ");
         int a = System.Convert.ToInt32 (System.Console.ReadLine() );
      }
      catch ( Exception  e )
      {  System.Console.WriteLine(" Ha habido un error..." + e.Message);
      }
      finally
      {  System.Console.WriteLine(" Con error y Sin error, este mensaje aparece. ");
         System.Console.ReadLine();
      }
    }
```

Manejo de cadenas de caracteres

```
string nombre = "Juan";
```

Para conocer la longitud (tamaño) de la cadena:

```
nombre.Lenght
```

Para convertir la cadena a minúsculas:

```
nombre.ToLower( )
```

Para convertir la cadena a mayúsculas:

```
nombre.ToUpper( )
```

Para obtener el tercer caracter de la cadena:

```
nombre.Substring(2,1)
```

Cuantos caracteres se desea obtener

Posicion del caracter deseado
(El primer caracter es la posición 0)

Comparación de cadenas (Manera 1)

```
string nombre1 = "Juan";
string nombre2 = "Maria";

if ( nombre1 == nombre2 )
{    //Qué hacer si son iguales
}
else
{    //Qué hacer si son diferentes
}
```

Comparación de cadenas (Manera 2)

```
string nombre1 = "Juan";
string nombre2 = "Maria";

if ( nombre1.Equals(nombre2) )
{    //Qué hacer si son iguales
}
else
{    //Qué hacer si son diferentes
}
```

Comparación de cadenas (Manera 3)

```
string nombre1 = "Juan";
string nombre2 = "Maria";

if (nombre1.CompareTo(nombre2) == 0)
{    //Qué hacer si son iguales
}
else
{    //Qué hacer si son diferentes
}
```

Vectores

```
//Declaracion en dos pasos
     int[ ] numeros;
     numeros = new int[ ] { 5, 10, 15, 20, 25};

//Declaracion en un paso
     int[ ] numeros2 = new int[ ] { 5, 10, 15, 20, 25 };

//Sin declaracion de elementos
     int[ ] numeros3 = new int[5];
     numeros3[0] = 5;
     numeros3[1] = 10;
     numeros3[2] = 15;
     numeros3[3] = 20;
     numeros3[4] = 25;
```

Uso del foreach en colecciones

```
foreach (int x in numeros)
{
 System.Console.WriteLine ( x );
}
```

Realiza un recorrido POR CADA elemento del vector "numeros" y lo imprime en pantalla. Nótese que no es necesario conocer el numero de elementos del vector para realizar el ciclo.

Matrices

```
//Declaracion en dos pasos
   int[ , ] matriz;
   matriz = new int[ , ] { { 5, 10 }, { 15, 20 } };

//Declaracion en un paso
    int[ , ] matriz2 = new int[ , ] { { 5, 10 }, { 15, 20 }};

//Sin declaracion de elementos
      int[ , ] matriz3 = new int[ 2,2 ];
      matriz3[0,0] = 5;
      matriz3[0,1] = 10;
      matriz3[1,0] = 15;
      matriz3[1,1] = 20;
```