C# DESDE EL PUNTO DE VISTA DE C

C# es un lenguaje de programación bastante semejante en lo que a sintaxis se refiere a C++. Por otra parte, C++ puede considerarse como un superconjunto de C.

En este documento analizaremos diversos ejemplos de programación utilizados habitualmente en C, y su correspondiente codificación en C#.

Por lo tanto, este documento no pretende ser un estudio exhaustivo de los tipos de datos, estructuras de control y librerías de C#, sino una muestra de un conjunto de ejemplos de código escritos en ambos lenguajes. No se trata entonces de justificar todo cuanto se escribe, con cuidado de haber dado previamente las definiciones necesarias, sino de mostrar de forma intuitiva las diferencias sintácticas de ambos lenguajes en aquellos aspectos en que nos encontramos familiarizados en C. En otras palabras, con esto se quiere justificar la posible falta de rigor expositivo.

Todo lo que aquí se muestra, se muestra de forma muy simple, procurando huir de elementos muy diferenciales. La forma de proceder utilizada en C# para estos ejemplos no es la más adecuada, sino la más aproximada a C, sin exponer en ningún caso otras alternativas mejores disponibles.

Se parte de la base de que el alumno "sabe C", y este documento es una primera toma de contacto con el nuevo lenguaje partiendo de ese supuesto. Por lo tanto, si el alumno no comprende los ejemplos propuestos en C, debería repasar "un poco" los contenidos de las asignaturas previas.

Para simplificar el código al máximo y quedarnos con lo esencial en lo que a sintaxis se refiere, todos los ejemplos están realizados en modo consola, tanto en C como en C#.

1. Hola mundo.

Hola mundo en C

```
#include <stdio.h>

void main (int argc, char* argv[])
{
    printf ("Hola mundo\n");
}
```

Hola mundo en C#

```
using System;

namespace Simple
{
        class CSimple
        {
                [STAThread]
                static void Main(string[] args)
                {
                   Console.Write ("Hola mundo\n");
                }
        }
}
```

El ejemplo de la izquierda muestra un típico programa 'Hola mundo' en C. Es necesaria la directiva `#include` para poder usar la librería que define la función `printf`. La función `main` recibe el número de argumentos pasados por la línea de comandos, y el array de cadenas de caracteres que los contiene.

El ejemplo de la derecha muestra un típico programa 'Hola mundo' escrito en C#. Es necesaria la línea `using System` para acceder a la clase `Console` y al método `Write`. La función `Main` en este caso recibe directamente un array de `strings` (cadenas de caracteres en C#), como en el caso anterior. El número de argumentos está implícito en el array de C#.

En C#, además, es necesario un espacio de nombres (`namespace Simple`) y una clase (`class CSimple`), para poder contener la función `Main`, que será el punto de entrada del programa al ejecutarse.

Desde un punto de vista práctico y, si se quiere, simplista, dentro de la clase `CSimple` pondremos todo lo que pondríamos en el fichero fuente de C a excepción de las directivas `include` y `define`.

2. Declaración de variables elementales e impresión de información por pantalla.

Ejemplo en C

```
#include <stdio.h>

void main (int argc, char* argv[])
{
    int i=4;
    double k=3.74;
    char C[] = "Hola que tal";

    printf ("i = %d, k = %f, C = %s\n",i,k,C);
}
```

Ejemplo en C#

```
using System;

namespace Simple
{
      class CSimple
      {
          [STAThread]
          static void Main(string[] args)
          {
            int i = 4;
            double k = 3.74;
            string C = "Hola que tal";

            Console.Write ("i = {0}, k = {1}, C = {2}\n",i,k,C);
            Console.WriteLine ("i = " + i + ", k = " + k + ", C = " + C);
          }
      }
}
```

Vemos que las declaraciones de variables (por ahora) son semejantes en ambos lenguajes. Se ha declarado en C un array de caracteres, aunque no es un tipo simple, para comprobar su equivalencia con el tipo `string` de C#.

En C# se ha hecho la impresión, usando el método `Write` de `Console`, de dos formas diferentes: La primera usa `{0}`, `{1}`, etc. para indicar los argumentos a imprimir, mostrando su semejanza con los `%d` y `%f`, de C. La segunda usa el operador de concatenación de strings (+), para unir los fragmentos que se desean imprimir.

Como vemos, las variables se declaran en C# de forma semejante a C, y el método `Console.Write` es el equivalente al `printf`, con algunas diferencias de formato.

3. Entrada de datos por la consola (teclado). Conversión de tipos.

Ejemplo en C

```
#include <stdio.h>

void main (int argc, char* argv[])
{
    char  Cadena[256];
    int   DatoEntero;
    float DatoReal;

    printf ("Introduce tu nombre: ");
    scanf ("%s",Cadena);

    printf ("Introduce tu edad: ");
    scanf ("%d",&DatoEntero);

    printf ("Introduce el precio de una barra de pan: ");
    scanf ("%f",&DatoReal);

    printf ("Tu nombre: %s\n",Cadena);
    printf ("Tu edad:   %d\n",DatoEntero);
    printf ("Barra pan: %f\n",DatoReal);
}
```

Ejemplo en C#

```
using System;

namespace Simple
{
      class CSimple
      {
            [STAThread]
            static void Main(string[] args)
            {
            string Cadena;
            int DatoEntero;
            double DatoReal;

            Console.Write ("Introduce tu nombre: ");
            Cadena = Console.ReadLine();
            Console.Write ("Introduce tu edad: ");
            DatoEntero = Convert.ToInt32 (Console.ReadLine());
            Console.Write ("Introduce el precio de una barra de pan: ");
            DatoReal = Convert.ToDouble (Console.ReadLine ());

            Console.WriteLine ("Tu nombre: " + Cadena);
            Console.WriteLine ("Tu edad:   " + DatoEntero.ToString());
            Console.WriteLine ("Barra pan: " + DatoReal.ToString("0.00"));
            }
      }
}
```

Vemos que toda la entrada por la consola se realiza a través de la función `ReadLine` de `Console`. `ReadLine` lee hasta que llega un <CR> (ENTER) del teclado, y retorna siempre un `string`. Es como si utilizáramos `scanf ("%s",Cad)` siempre, y luego convirtiéramos la cadena de caracteres (con las funciones `atoi` y `atof`).

Las operaciones de la clase `Convert` permiten convertir los strings leídos por el teclado al tipo de datos deseado.

4. Operadores.

En general, los operadores de C son semejantes a los de C#. Tenemos la suma, la resta, el producto, la división (+, -, *, /, %), también operadores de comparación (==, !=, >, …), operadores de bits, etc. La siguiente tabla resume los operadores de C#.

Categorías	Operadores
Aritméticos	`+ - * / %`
Lógicos (booleanos y bit a bit)	`& | ^ ! ~ && || false true`
Concatenación de cadenas	`+`
Incremento y decremento	`++ --`
Desplazamiento	`<< >>`
Relacionales	`== != < > <= >=`
Asignación	`= += -= *= /= %= &= |= ^= <<= >>=`
Acceso a miembros	`.`
Indización	`[]`
Conversión de tipos explícita	`()`
Condicional	`?:`
Concatenación y eliminación de delegados	`+ -`
Creación de objetos	`new`
Información de tipos	`is sizeof typeof`
Control de excepciones de desbordamiento	`checked unchecked`
Direccionamiento indirecto y dirección	`* -> [] &`

5. Sentencias de control y palabras reservadas.

Las sentencias de control también son básicamente las mismas. C# tiene algunas más, que ya iremos viendo, pero por ahora no son relevantes.

Un `if` se escribe igual en C que en C#, un `for` y un `while` también, etc. Por el momento no debemos prestarle más atención a este tema, aunque debemos saber que C# es más potente que C en este aspecto.

La lista de palabras reservadas de C# es la siguiente:

abstract	**event**	**new**	struct
as	**explicit**	**null**	switch
base	extern	**object**	**this**
bool	**false**	**operator**	**throw**
break	**finally**	**out**	**true**
byte	**fixed**	**override**	**try**
case	float	**params**	**typeof**
catch	for	**private**	**uint**
char	**foreach**	**protected**	**ulong**
checked	goto	**public**	**unchecked**
class	if	**readonly**	**unsafe**
const	**implicit**	**ref**	**ushort**
continue	**in**	return	**using**
decimal	int	**sbyte**	**virtual**
default	**interface**	**sealed**	**volatile**
delegate	**internal**	short	void
do	**is**	sizeof	while
double	**lock**	**stackalloc**	
else	long	static	
enum	**namespace**	**string**	