C# FOR BEGINNERS

**C# vs NET**

**C#** es un lenguaje de programación.

**NET** es un framework para construir aplicaciones en Windows. Consta de 2 componentes:

* CLR (Common Language Runtime): es una aplicación que se encuentra en la memoria, cuyo trabajo es traducir el código dentro del código de la máquina. Esto se le llama **Just-in-Time Compilation (JIT)**.
* Class library.

**OUR FIRST C# APPLICATION**

Creamos un proyecto de aplicación de consola, y con **ctrl F5** activamos la consola.

Un **namespace** es un contendor para las **class**.

Un **assembly** es un archivo, en forma de ejecutable o DLL, que contiene uno o más namespaces y classes.

**VARIABLES and CONSTANTS**

Para declarar variables primero necesitamos pasarle que tipo de datos es, luego el nombre de la variable y por último su valor.

Para declarar variables constantes, necesitamos utilizar la palabra clave **const**, el tipo de dato, el nombre y luego el valor.

**int** number = 1;

**const float** pi = 3.14;

**Identificadores:**

* No pueden empezar con un número: **1route** (mal), **oneRoute** (bien).
* No pueden incluir un espacio en blanco: **first name**(mal), **firstName**(bien).
* No puede ser una palabra clave reservada: **int** (mal), podemos usar una arroba en cambio **@int.**
* Usar nombres descriptivos: **fn** (mal), **firstName** (bien).

**Datos primitivos:**

* Byte.
* Short.
* Int.
* Long.
* Float.
* Double.
* Decimal.
* Char.
* Bool.

**Datos no primitivos:**

* String.
* Array.
* Enum.
* Class.

**OVERFLOWING**

Es cuando excedemos el límite del tipo de datos de bytes en C# por defecto. Por ejemplo:

**byte** number = 255;

number = number + 1; **// 0**

El resultado de esto es 0 porque excedimos el tipo de datos en bytes.

Para evitarlo podemos usar **checked**: esto nos va a dar un error para evitar que todo se rompa.

**Checked** {

**byte** number = 255;

number = number + 1;

}

**DEMO: Variables and Constants**

Para compilar la aplicación vamos a utilizar **Ctrl + Shift + B**.

Para correr la aplicación vamos a utilizar: **Ctrl + F5**.

byte number = 2;

int count = 10;

float totalPrice = 20.95f;

char character = 'a';

string firstName = "Joaquin";

bool isWorking = true;

Podemos cambiar todos los tipos de datos por **var** y los va a reconocer igual:

var number = 2;

var count = 10;

var totalPrice = 20.95f;

var character = 'a';

var firstName = "Joaquin";

var isWorking = true;

**Template:**

Console.WriteLine("{0} {1}", byte.MinValue, byte.MaxValue);

Da como resultado el valor mínimo y máximo de un byte.

**TYPE CONVERSION**

**Implicit Type Conversion:** si esto estaría al revés se perdería data y no se puede copilar.

byte b = 1;

int i = b;

**Explicit Type Conversion:** aca debemos especificarle al copilador que estamos advertidos de que se puede perder data, pero lo hacemos igualmente.

Float f = 1.0f;

Int i = (int)f;

**Non-compatible types:**

String s = “1”;

Int i = **Convert.toInt32(**s**);**

Int j = int.Parse(s);

**Convert:**

* ToByte().
* ToInt16().
* ToInt32().
* ToInt64().

**Try and Catch:** si queremos pasar un string por ejemplo a un byte y este string es más grande de lo que puede soportar el dato bytes (255) hay que hacer un bloque try and catch.

try

{

var number = "1234";

byte b = Convert.ToByte(number);

Console.WriteLine(b);

}

catch (Exception)

{

Console.WriteLine("No se puede convertir en un byte.");

}

**OPERATORS**

Los mismos operadores que en JavaScript.

var a = 10;

var b = 3;

Console.WriteLine((float)a / (float)b);//3,333

**NON-PRIMITIVES TYPES**

**CLASSES**

Para crear una clase utilizamos un modificador de acceso(**public**), la palabra reservada **class** y el nombre en mayúscula que elijamos.

public class Calculator

{

public int Add(int a, int b)

{

return a + b;

}

}

Si queremos acceder al método **Add** de la **class** Calculator, podemos especificarle un **static**:

Public **static** int Add(){}

Y luego accedemos a ella sin tener que crear un objeto:

Int result = Caluculator.Add(1, 2);

Podemos crear una carpeta que almacene las clases dependiendo su característica, como puede ser una carpeta llamada **Math** que contenga una **class** llamada Calculator, luego debemos crear ese archivo dentro de la carpeta que contenga la clase.

**Archivo calculator dentro de Math:**

namespace HelloWorld.Math

{

public class Calculator

{

public int Add(int a, int b)

{

return a + b;

}

}

}

**Dentro del archivo principal:**

using HelloWorld.Math;

Calculator calculator = new Calculator();

var result = calculator.Add(1, 2);

Console.WriteLine(result);

**STRUCTS**

Para declarar un **struct** es igual que una **class** pero cambiando estas palabras claves.

Utilicemoslo cuando necesitemos crear cosas sencillas.

**ARRAYS**

int[] numbers = new int[3]; //{1, 2, 3};//int[3] significa cuantos elementos le pasamos al array

numbers[0] = 1;

numbers[1] = 2;

numbers[2] = 3;

Podemos pasarle los datos de dos formas con las curly braces después de decirle cuantos elementos tiene, o numbers[0] = 1, por ejemplo.