**Evaluation Only. Created with Aspose.Words. Copyright 2003-2021 Aspose Pty Ltd.**

Tipos de datos



**Nombre Rango**

sbyte De -128 a 127

byte De 0 a 255

short De -32 768 a 32 767

ushort De 0 a 65.535

int De -2.147.483.648 a 2.147.483.647

uint De 0 a 4.294.967.295

long De -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807 ulong De 0 a 18.446.744.073.709.551.615

float 7 dígitos

double 15-16 dígitos

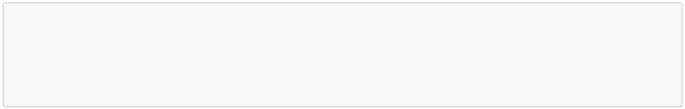
decimal 28-29 dígitos significativos

char Carácter entre comillas simples ' '

bool Booleano

var es un "comodín" que se usa para no tener que indicar el objeto que te viene de vuelta (un string, un int, un float, un obj...)

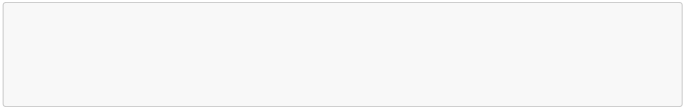
// [tipo de dato] [NombreVariable];



// [tipoDato] [NombreVariable] = [Valor de la variable]; byte x = 0;

No todos los tipos de datos se pueden inicializar a null como por ejemplo los Enum, los int, long, etc. Si necesitamos que puedan ser nulos esos, habrá que declararlos como tipo Nullable<T>, ejemplo:

// Las dos líneas hacen exactamente lo mismo Nullable<int> nullableUno = null;

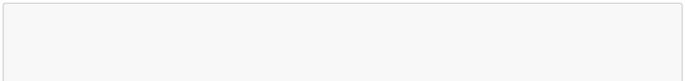


int? nullableDos = null;

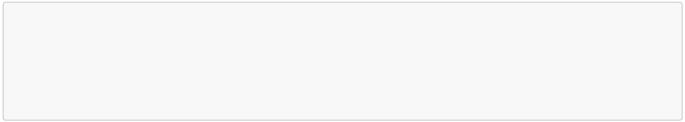
Cadenas(String)

Los string son un conjunto de caracteres(char) que se reflejan con dobles comillas ""

1 / 4



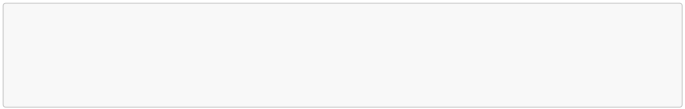
2\_TiposAndSentenciasDeFlujo.md 1/21/2020



string raw = "soy una cadena o string"; Console.WriteLine("Esto es un método que recibe un string");

Para rutas se puede usar @""

string raw = @"C\git"; string masLineas = @"hola Adios";



Una forma de concatenas strings con variables es:

string concatenando = $"{variable1} {variable2}, etc.";



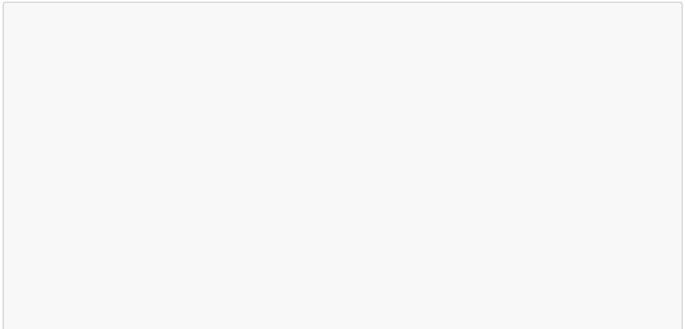
Métodos habituales en cadenas

* cadena.Replace(x, y) -> Devuelve una cadena en la que se reemplazan las letras o asi metidas en "x" por las de "y"
  + cadena.Split(x) -> Devuelve un Array con la cadena separada dividiéndola cada vez que encuentre el char enviado, por defecto sera el símbolo '-'
* cadena.Remove(x) -> Devuelve una cadena con los elementos de "x" eliminados, si esta vacío eliminara espacios
  + cadena.StartsWith(x) -> Devuelve true o false si inicia la cadena por "x", usado mucho en menus de opciones

Sentencias de flujo



if (a == b || b == c && !d){



Console.WriteLine("pasa por verdadero"); } else if ( a != b){

Console.WriteLine("diferente");

} else{

Console.WriteLine("pues nah!");

}

switch(b3) {

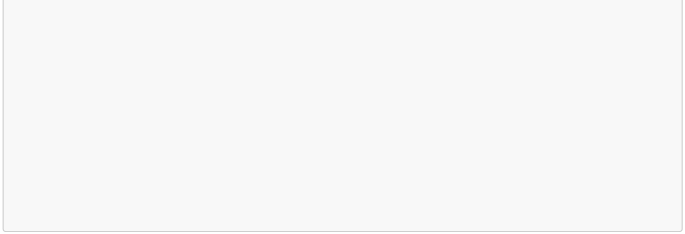
case 0:

Console.WriteLine("es 0"); break;

case 1:

Console.WriteLine("es 1");

2\_TiposAndSentenciasDeFlujo.md 1/21/2020



break;

case 2:

Console.WriteLine("es 2"); break;

case 3:

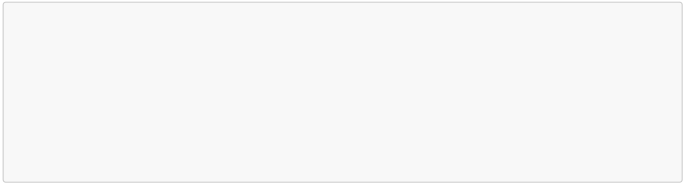
Console.WriteLine("es 3"); break;

}

Operador ternario

Se usa para no tener que hacer comparaciones cortas para agregar un dato u otro por ejemplo, es un if- else

string a3 = "0";



int b3 = 2;

string ternario = a3.Equals("0") ? a3 : b3.ToString(); // if a3.Equals("0")

// ternario = a3;

// else

// ternario = b3.ToString();

Tambien hay otro operando que se usa para comprobar el valor NULL.

string a3 = null;



int b3 = 2;

string comprobacion = a3 ?? b3; // if a3 != null

// comprobacion = a3

// else

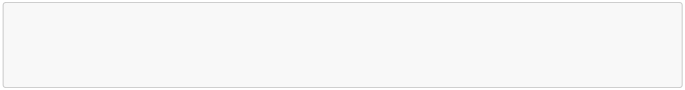
// comprobacion = b3

* En el lado Izquierdo del ?? se pondrá el valor a compara si es o no null y si no lo es, sera el valor que se agregará
  + En la zona Derecha del ?? se pondrá un valor por defecto en el caso de que el valor izquierdo sea NULL

Operadores condicionales NULL ?. y ?[]

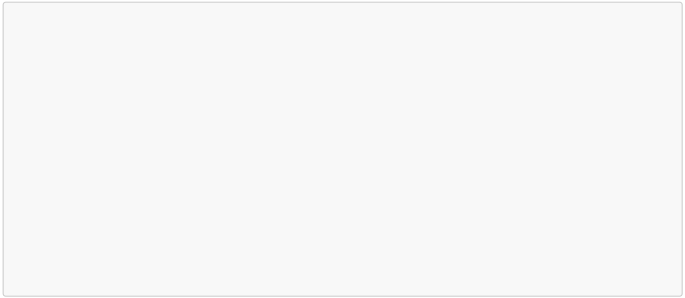
si una operación en una cadena de la operación de acceso a elementos o miembros condicional devuelve null, no se ejecuta el resto de la cadena. En el ejemplo siguiente, B no se evalúa si A se evalúa como null y C no se evalúa si A o B se evalúan como null:

A?.B?.Do(C); A?.B?[C];



Bucles

int edad = 0;



while (edad < 3) {

edad ++;

Console.WriteLine($"{edad}"); }

for(int x = 0; x < 3; x++){

edad ++;

Console.WriteLine($"{edad} rango: {x}"); }

foreach (var i in lista)

Console.WriteLine($"{i}");