



# TEMA 1 FUNDAMENTOS DE PROGRAMACIÓN

## PARTE 4

Msc. Ing. Joel Reynaldo Alánez Durán



**PALABRAS RESERVADAS**

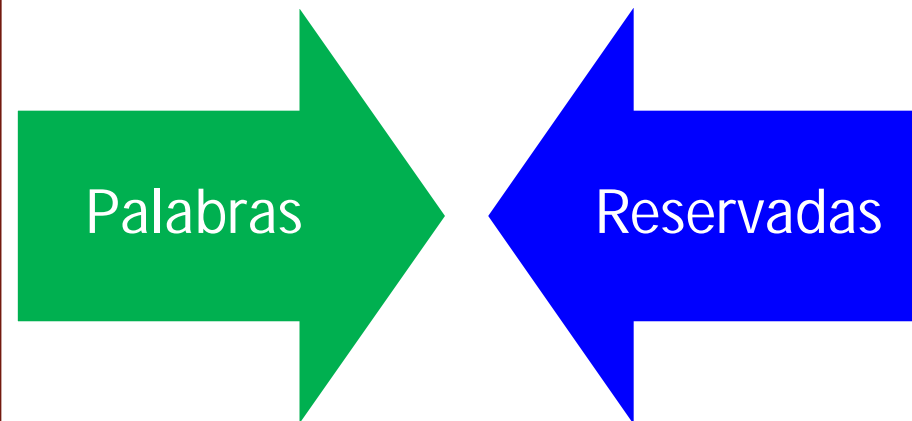
# Palabras Reservadas: C#

Las palabras reservadas o claves son identificadores predefinidos que tienen un significado especial para el compilador.

No se pueden usar como identificadores en un programa a menos que incluyan el carácter @ como prefijo.

Ej: `int @if;` // Es válido

Dentro del código que se escribe en Visual Studio las palabras reservadas aparecen de color azul.



abstract

break

char

continue

do

evento

finally

foreach

in

internal

Espacio de nombres

':?'

params

readonly

sealed

static

this

typeof

unsafe

void

as

byte

checked

decimal

double

explicit

fixed

goto

in

is

new

out

private

ref

short

string

throw

uint

ushort

volatile

base

case

class

default

else

extern

float

if

int

lock

null

out

protected

return

sizeof

struct

true

ulong

using

while

bool

catch

const

delegate

enum

false

for

implicit

interfaz

long

object

override

public

sbyte

stackalloc

switch

try

unchecked

virtual



# TIPOS DE DATOS

# Tipos de Datos C#: Enteros

C# Tipo	.Net Framework Tipo	Bytes en Ram	Rango
sbyte	System.Sbyte	1	-128 a 127
short	System.Int16	2	-32768 a 32767
int	System.Int32	4	-2147483648 a 2147483647
long	System.Int64	8	-9223372036854775808 a 9223372036854775807
byte	System.Byte	1	0 a 255
ushort	System.UInt16	2	0 a 65535
uint	System.UInt32	4	0 a 4294967295
ulong	System.UInt64	8	0 a 18446744073709551615

## Tipos de Datos C#: Reales

C# Tipo	.Net Framework Tipo	Bytes en Ram	Rango
float	System.Single	4	Aprox. $\pm 1.5 \times 10^{-45}$ a $\pm 3.4 \times 10^{38}$ con 7 decimales
double	System.Double	8	Aprox. $\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$ con 15 o 16 decimales
decimal	System.Decimal	12	Aprox. $\pm 1.0 \times 10^{-28}$ a $\pm 7.9 \times 10^{28}$ con 28 o 29 decimales

## Tipos de Datos C#: **Caracter**

C# Tipo	.Net Framework Tipo	Bytes en Ram	Rango
char	System.Char	2	Cualquier caracter Unicode





# Tipos de Datos C#: Cadenas de Caracteres

C# Tipo	.Net Framework Tipo	Tipo
string	System.String	Cadena de Caracteres



## Tipos de Datos C#: Lógico (Booleano)

C# Tipo	.Net Framework Tipo	Bytes en Ram	Rango
bool	System.Boolean	1/2	true o false

# Tipos de Datos: C#

Los tipos de datos mas usados son:

- Enteros: `int`
- Reales: `double`
- Carácter: `char`
- Cadenas de Caracteres: `string`
- Lógicos: `bool`





# COMENTARIOS

## ■ Comentarios: Definición

- Son anotaciones legibles al programador en el código fuente de un Programa informático.
  - Estas anotaciones son potencialmente significativas para los programadores, pero usualmente ignorados por los compiladores e intérpretes.
- Los comentarios son añadidos usualmente con el propósito de hacer el código fuente más fácil de entender.

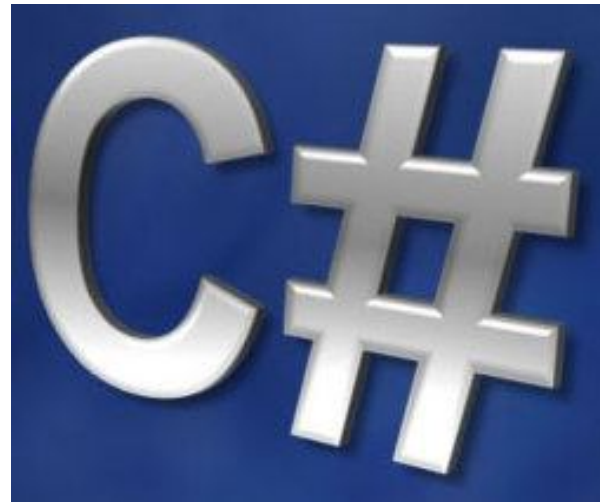
## Comentarios: C#

Para comentar una sola línea.

```
//Tu comentario AQUÍ
```

Para comentar un Bloque de Texto

```
/* Tu bloque de Texto  
   AQUÍ  
*/
```





# VARIABLES

# Variables: C#

Las variables son declaradas usando la siguiente forma general:

`tipo nombre_variable;`

Donde:

tipo → es el tipo de dato de la variable.

nombre\_variable → es el identificador .

Es importante saber que las capacidades de la variable son determinadas por su tipo.

Ejemplo una variable de tipo bool(Logico) no puede ser usado para guardar un entero(int)

Ejemplo:

```
int cont;  
cont = 1; //Esto compila
```

Ejemplo 2:

```
bool x;  
x=2; //Esto arroja un error
```



# Variables: C#

Todas las variables deben ser declaradas. Esto es necesario porque el compilador debe saber que tipo de dato contiene una variable, de tal forma que pueda compilar cualquier sentencia que usa una variable.

C# define varios diferentes tipos de variables.

Los tipos que se usaran con mas frecuencia son las llamadas «variables locales» por que son declaradas dentro de un método.



# ■ Inicializando Variables: C#

Una manera de dar un valor a una variable es a través de una sentencia de asignación. Otra es dando un valor inicial cuando es declarada.

Forma general:

`tipo nombre_variable =  
valor;`

El valor debe ser compatible con el tipo especificado.

```
//Dar a suma un valor inicial de 10;  
int suma=0;  
// inicializa digito con la letra A  
char digito ='A';  
//sueldo es inicializado con 5500.5  
double sueldo = 5500.5;
```

## Variables: C#

Cuando se declaran 2 o mas variables del mismo tipo, es posible usar una lista separada por comas donde es posible asignar uno o mas valores iniciales a las variables

```
// b y c tienen inicializaciones  
int a, b = 8, c = 19, d;
```

# Variables C#: Inicialización Dinámica

C# permite inicializar una variable de forma dinámica usando cualquier expresión válida en el punto en el cual la variable es declarada.

```
//Hallar el volumen de un cilindro:  
double radio = 4, altura = 5;
```

```
//Dinamicamente se inicializa  
//el volumen
```

```
double volumen = 3.1416*radio*radio*altura;  
MessageBox.Show("Volumen es " + volumen);
```

# Variables de Tipo Implícito: C#

Normalmente la declaración de una variable incluye el tipo de variable como int, bool, double, etc seguido del nombre de la variable.

En c# es posible que el compilador determine el tipo de una variable basado en el valor usado al momento de inicializar dicha variable.

A esto se conoce como variable de tipo implícito.

Una variable de tipo implícito es declarado por la palabra reservada **var** y este debe ser inicializado.

```
// Estas son variables de tipo implicito  
var pi = 3.1416; // pi es double  
var radio = 10; // radio es int
```

```
// msg es una cadena(string).  
var msg = "Radius: ";
```

```
// Explicito declarado ares como double  
double area;  
area = pi * radio * radio;
```

```
//La siguiente sentencia no será  
//compilada porque el radio es un entero  
//(int) y no puede ser asignado como  
//real(double)  
radio = 12.2; // Error!
```



# CONSTANTES

# ■ Constantes: C#

Las constantes son usadas para referir identificadores cuyos valores no cambiarán (el valor permanecerá fijo) durante la ejecución de todo el programa.

Es importante diferenciar entre una constante y una variable, ya que las constantes ocupan menos memoria y ayudan al rendimiento del programa.

Para declarar una constante se sigue la siguiente sintaxis:

```
const tipo NOMBRE_CONSTANTE;
```

Por convención el nombre de la constante se escribe con mayúsculas:

//Ejemplos

```
const double PI = 3.1416;
```

```
const string msg = "ERROR";
```



# OPERADORES Y EXPRESIONES



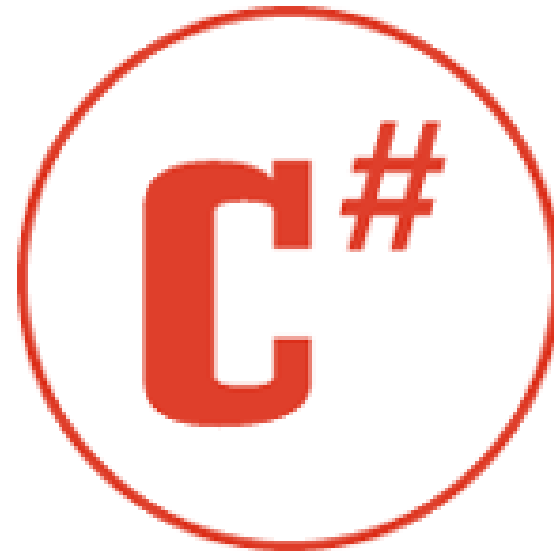
# ■ Expresiones: C#

Se le llama expresiones a aquellas relaciones que una vez evaluadas producen un resultado. Toda expresión tiene operandos y operadores.

- **Operandos:** Valores que los operadores manipulan. Son identificadores o valores literales (números).
- **Operadores:** Trabajan sobre los operandos.

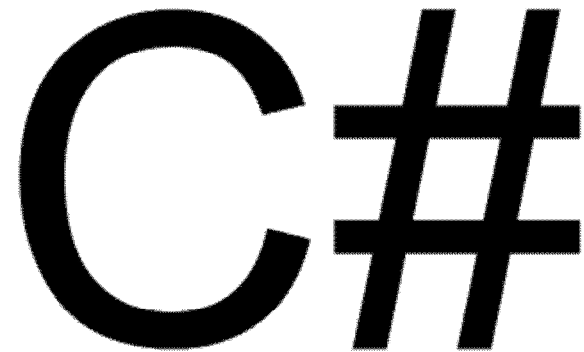
**Ejemplo de expresión:**

A=2+B



# Operadores en C#: Tipos

- C# tiene 4 clases de operadores:  
aritméticos(arithmetic), bit a bit(bitwise),  
relacionales(relational) y lógicos(logical).
- También tiene varios otros operadores que se manejan en situaciones especiales.
- En este capítulo se verán los operadores aritméticos, relacionales y lógicos.

A large, stylized logo for C# programming language, featuring a bold 'C' followed by a sharp '#' symbol.

# Operadores Aritméticos

Operador	Significado
+	Adición
-	Sustracción
*	Multiplicación
/	División (Para Enteros es una división exacta)
%	Modulo
++	Incremento
--	decremento

# ■ Expresiones Aritméticas

$A * B$

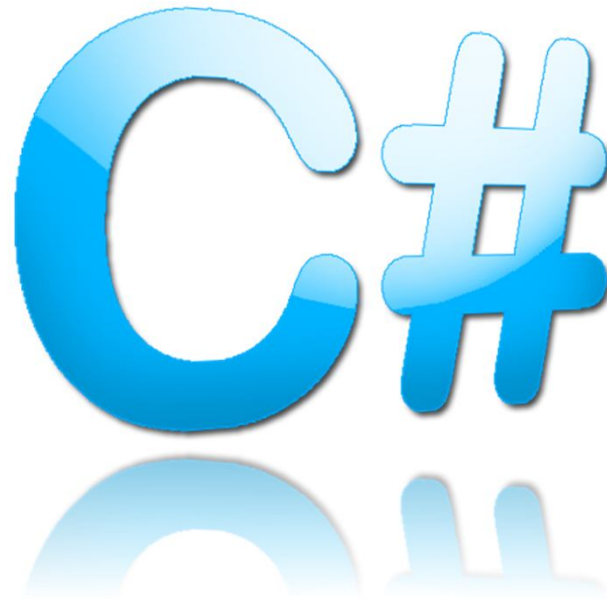
$A + B$

$A - B$

$A / B$

$A \% B$

$((A * B) + C)$



# Expresiones Aritméticas: Ejemplo

```
int iresult, iresto;  
double dresult, dresto;  
iresult = 10 / 3;  
iresto = 10 % 3;  
dresult = 10.0 / 3.0;  
dresto = 10.0 % 3.0;  
MessageBox.Show("El resultado y el resto de 10 / 3: " + iresult + " " + iresto);  
MessageBox.Show("El resultado y el resto de 10.0 / 3.0: " + dresult + " " + dresto);
```



# Incremento y decremento

Los incrementos se manejan de dos formas

- Postfijo ( $x=x+1$ ) , es lo mismo que  $x++$
- Prefijo  $++x$ ;

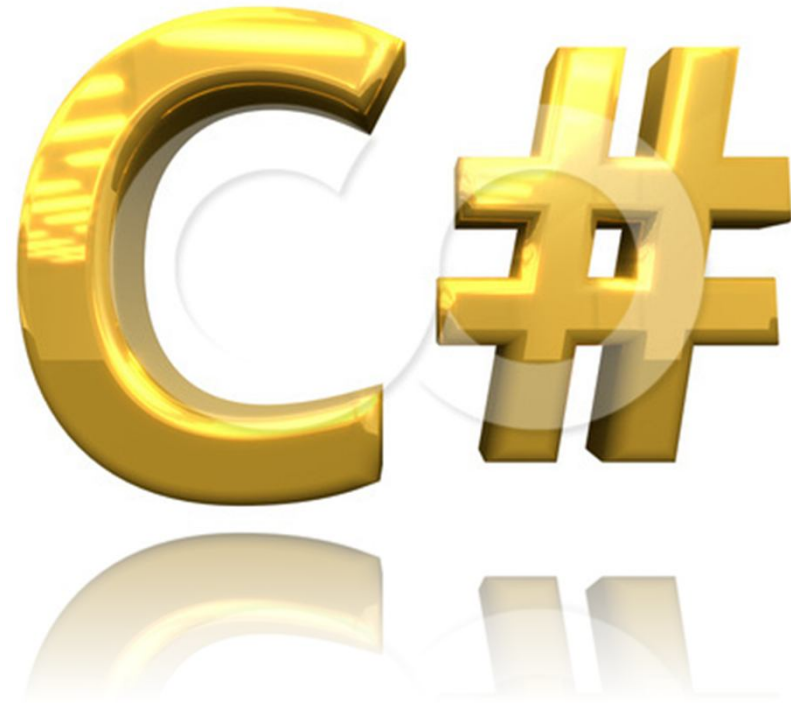
Lo propio ocurre con el decremento

- Postfijo: ( $x=x-1$ ); es lo mismo que  $x--$
- Prefijo  $--x$

Existe una diferencia importante entre prefijo y postfijo cuando se trabaja con expresiones.

En el caso de Prefijo el valor de la variable se incrementa o decrementa y luego se asigna su valor.

En el caso de postfijo el valor de la variable es primero asignado y luego es incrementado o decrementado.



# Incremento y decremento: Ejemplo

Caso 1

```
int x = 10;
```

```
int y;
```

```
y = ++x;
```

En este caso y tendrá 11, esto por que x es incrementado primero y luego es obtenido su valor

Caso2

```
int x = 10;
```

```
int y;
```

```
y = x++;
```

En este caso y contiene 10, esto pro que el valor de x es primero obtenido y entonces el original del valor de x es retornado.



# ■ Operadores Relacionales y Lógicos

Las relaciones que los valores pueden tener entre si son controladas por los operadores relacionales.

Los operadores lógicos definen si las relaciones arrojan resultados verdaderos o falsos.

■ Los operadores relacionales producen resultados verdadero o falso y trabajan con operadores lógicos.





# Operadores Relacionales

Operador	Significado
==	Igual a
!=	Diferente de
>	Mayor que
<	Menor que
>=	Mayor igual que
<=	Menor igual que

# Operadores Lógicos

Operador	Significado
&	AND
	OR
^	XOR(exclusive OR)
	corto-circuito OR *
&&	corto-circuito AND *
!	NOT

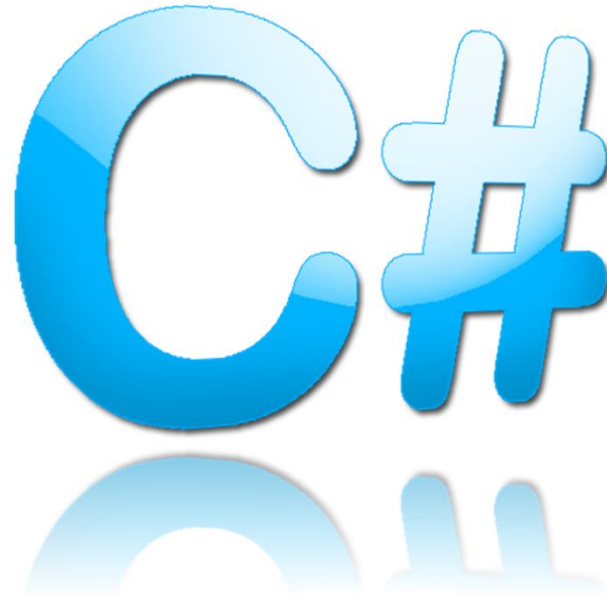
# Expresiones Lógicas

$d \neq 0 \ \&\& \ (n \% d) == 0$

$B < C$

$A > F \ \&\& \ A \neq 0$

$A > F \ || \ A \neq 0$



# Operadores Relacionales y Lógicos

- El resultado de los operadores relacionales y lógicos es un valor bool.
- En general, los objetos pueden ser comparados para igualdad o desigualdad usando `==` y `!=`.
- También se puede comparar con los operadores `<`, `>`, `<=`, `>=`.



# Operadores Relacionales y Lógicos:

## Ejemplo

```
int i, j;  
bool b1, b2;  
i = 10;  
j = 11;  
if (i < j) MessageBox.Show("i < j");  
if (i <= j) MessageBox.Show("i <= j");  
if (i != j) MessageBox.Show("i != j");  
if (i == j) MessageBox.Show("Esto no se ejecutara");  
if (i >= j) MessageBox.Show("Esto no se ejecutara");  
if (i > j) MessageBox.Show("Esto no se ejecutara");  
b1 = true;  
b2 = false;  
if (b1 && b2) MessageBox.Show("Esto no se ejecutara");  
if (!(b1 && b2)) MessageBox.Show("!(b1 && b2) es verdadero");  
if (b1 || b2) MessageBox.Show("b1 || b2 es verdadero");  
if (b1 ^ b2) MessageBox.Show("b1 ^ b2 es verdadero");
```

# Operadores lógicos: Corto-Circuito

Son usados para producir un código mas eficiente. Para AND && y para OR ||

La única diferencia entre la normal (&, |) y corto-circuito (&&, ||) es que el normal siempre evalúan cada operador y corto-circuito se evalúa el segundo operando solo si es necesario.



# ■ Corto Circuito: Ejemplo

```
int n, d;  
n = 10;  
d = 2;  
// Aquí, d es 2, entonces la operación modulo se lleva acabo.  
if (d != 0 && (n % d) == 0)  
    MessageBox.Show (d + " es un factor de " + n);  
d = 0; // ahora, poner de en cero  
// Entonces si d es cero, el segundo operador no es evaluado.  
if (d != 0 && (n % d) == 0)//Expresión lógica  
    MessageBox.Show (d + " es un factor de " + n);  
// Ahora, intentar lo mismo sin corto-circuito.  
// Esto causa división entre cero y provocará error.  
if (d != 0 & (n % d) == 0)  
    MessageBox.Show (d + " es un factor de " + n);
```



LA CLASE MATH



## ■ La Clase Math

Es una clase que proporciona constantes y métodos estáticos para operaciones trigonométricas, logarítmicas y otras funciones matemáticas comunes.

**Math.NET**



# La Clase Math: Funciones

Nombre	Descripción
Abs	Devuelve el valor absoluto de un número.
Acos	Devuelve el ángulo cuyo coseno es el número especificado.
Asin	Devuelve el ángulo cuyo seno es el número especificado.
Atan	Devuelve el ángulo cuya tangente corresponde al número especificado.
Ceiling	Devuelve el valor integral más pequeño que es mayor o igual que el número decimal especificado.
Cos	Devuelve el coseno del ángulo especificado.
Cosh	Devuelve el coseno hiperbólico del ángulo especificado.
DivRem	Calcula el cociente de dos números enteros de 32 bits con signo y devuelve también el resto de la división como parámetro de salida.
Exp	Devuelve e elevado a la potencia especificada.
Floor	Devuelve el número entero más grande menor o igual que el número decimal especificado.
Log	Devuelve el logaritmo natural (en base e) de un número especificado.
Log10	Devuelve el logaritmo en base 10 de un número especificado.
Max	Devuelve el mayor de dos números
Min	Devuelve el menor de dos números



## La Clase Math: Funciones

Nombre	Descripción
Pow	Devuelve un número especificado elevado a la potencia especificada.
Round	Redondea un valor decimal al valor integral más próximo.
Sign	Devuelve un valor que indica el signo de un número.
Sin	Devuelve el seno del ángulo especificado.
Sinh	Devuelve el seno hiperbólico del ángulo especificado.
Sqrt	Devuelve la raíz cuadrada de un número especificado.
Tan	Devuelve la tangente del ángulo especificado.
Tanh	Devuelve la tangente hiperbólica del ángulo especificado.
Truncate	Calcula la parte entera de un número decimal especificado.

## La Clase Math: Ejemplo

EJEMPLO