

# Ejercicios Operadores

[Descargar estas soluciones](#)

## Índice

- [Ejercicio 1](#)
- [Ejercicio 2](#)
- [Ejercicio 3](#)
- [Ejercicio 4](#)
- [Ejercicio 5](#)
- [Ejercicio 6](#)
- [Ejercicio 7](#)
- [Ejercicio 8](#)
- [Ejercicio 9](#)
- [Ejercicio 10](#)

## Ejercicio 1

Indica cuales de los siguientes identificadores **no** son correctos y por qué.

- ✓ a) `contador` Correcto, porque sigue el convenio de **camelCasing**.
- ✓ b) `CONTADOR` Correcto, porque las **constantes** las podremos poner con todos los caracteres en mayúsculas.
- ✓ c) `_hola` Correcto, pero lo evitaremos en lo posible.
- ✗ d) `capacidad` Incorrecto, porque no esta claro, debería llevar unidades.
- ✗ e) `Ciudadan@` Incorrecto, no se puede usar el carácter "@" ya que es un **carácter del lenguaje**.
- ✗ f) `numVidas` Incorrecto, porque usa **contracciones**.
- ✓ g) `portal2` Correcto
- ✗ h) `2portal` Incorrecto, porque el número tiene que estar detrás de "portal".
- ✓ i) `SumaTotal` Correcto, porque sigue el convenio de **PascalCasing** aunque tiene algo de ambigüedad.
- ✓ j) `capacidad_cm3` Correcto.
- ✗ k) `Suma-Total` Incorrecto, porque no se puede usar el caracter - para separar palabras.
- ✗ l) `suma_total` Incorrecto, porque no sigue el convenio de .NET de no usar **snake\_casing** si no es una constante.

## Ejercicio 2

Indica cual de las siguientes definiciones de literales es incorrecta.

**Nota:** Puede serlo más de una.

- ✗ a) `uchar x = '\b';` Incorrecta, "**uchar**" no existe como tipo de variable.
- ✗ b) `char x = '\'\0';` Incorrecta, hay ambigüedad para que sea correcta tendría que ser `char x = '\0';`
- ✓ c) `ulong x = 456UL;` Correcta
- ✓ d) `double x = 0.0d;` Correcta
- ✗ e) `int x = 2L;` Incorrecta, correctamente sería 2 y no 2L.
- ✗ f) `float x = 3.2e-127f;` Incorrecta `e-127` supera el límite para simple precisión de `e-38`.
- ✗ g) `decimal x = 33.4;` Incorrecta, falta la **M** después de 33.4
- ✓ h) `string x = "";` Correcta, Estamos asignando la cadena vacía.
- ✗ i) `long x = 1000000.0L;` Incorrecta, No se puede convertir implícitamente el tipo "double" en "long".
- ✗ j) `string x = '\t\\"\n';` Incorrecta, habría que poner `"` al principio y al final.

## Ejercicio 3

¿Cómo definirías la constante alfanumérica siguiente?

```
'Hola'
¿Cómo te llamas?
```

- ✓ a) `""'Hola'\n¿Cómo te llamas?""` → Correcta
- ✗ b) `""'Hola'
¿Cómo te llamas?""` No se pueden poner saltos de línea al definir un literal de cadena.
- ✓ c) `""'\n'\"'\"'¿Cómo te llamas?""` → Correcta
- ✗ d) `""'Hola'\n¿Cómo te llamas\?'` No puedo definir un literal de cadena entre comillas simples.

## Ejercicio 4

Define en C# una variable que ocupe **1 byte** con signo e inicialízala en la declaración con el valor `000100002` en **hexadecimal**.

```
sbyte n = 0x10;
```

## Ejercicio 5

Indica cual sería el valor o posible error, de cada una de las siguientes variables.

```
int a = 3;           // a = 3
int b = null;        // Error de compilación
int c = default;     // c = 0
int? d = null;       // d = null
int? e = default;    // e = null
```

## Ejercicio 6

Sea el siguiente código, indica el valor de evaluar la expresión `i + j` al final del mismo.

Realiza una pequeña tabla con la **traza** de las variables después de ejecutar cada expresión.

```
int i = 2;
int j = 3;
int x = j = ++i * j++;
int y = x + i / j;
i = y % ++j;
```

|                               | i=2      | j=3       | x=?      | y=?      |
|-------------------------------|----------|-----------|----------|----------|
| <b>int x = j = ++i * j++;</b> |          |           |          |          |
| int x = j = 3 * 3;            | <b>3</b> | <b>4</b>  |          |          |
| int x = j = 9;                |          |           |          |          |
| 9;                            | 3        | <b>9</b>  | <b>9</b> | ?        |
| <b>int y = x + i / j;</b>     |          |           |          |          |
| int y = x + 3 / 9;            |          |           |          |          |
| int y = 9 + 0;                |          |           |          |          |
| 9;                            | 3        | 9         | 9        | <b>9</b> |
| <b>i = y % ++j;</b>           |          | <b>10</b> |          |          |
| i = y % 10;                   |          |           |          |          |
| i = 9 % 10;                   |          |           |          |          |

|    | <b>i=2</b> | <b>j=3</b> | <b>x=?</b> | <b>y=?</b> |
|----|------------|------------|------------|------------|
| 9; | <b>9</b>   | 10         | 9          | 9          |

**i + j** se evaluará a **19**

## Ejercicio 7

Indica cuales de las expresiones siguientes son verdaderas y cuales falsas, suponiendo que

`x = 20, y = 10, z = 5, w = 2, F = false, T = true`.

a) `x > y && z > w`

```
x > y && z > w
20 > 10 && 5 > 2
T && T
T
```

b) `x < y && z > w`

```
x < y && z > w
20 < 10 && 5 > 2
F && T
F
```

c) `x < y || z > w`

```
x < y || z > w
20 < 10 || 5 > 2
F || T
T
```

d) `!F`

```
!F
T
```

e) `!T`

```
!T
F
```

f) `!!F`

```
!!F
!T
F
```

g) **!(F == F)**

```
!(F==F)
!(T)
F
```

h) **10 > 5 && !(10 < 9) || 3 <= 4**

```
10 > 5 && !(10<9) || 3 <= 4
10 > 5 && !(F) || 3 <= 4
10 > 5 && T || 3 <= 4
T && T || T
T
```

i) **T && !F || T**

```
T && !F || T
T && T || T
T
```

j) **T && !(F || T)**

```
T && !(F || T)
T && !(T)
T && F
F
```

## Ejercicio 8

¿Cuál será el valor de la variable definida como `int r` en la siguiente expresión de asignación?

```
int r = (int)('C' - (float)5 / 2 + 3.5f + 0.4f);
```

**Nota:** Analiza la expresión **paso a paso** como hemos hecho en clase.

```
int result = (int) ('C' - (float)5 / 2 + 3.5f + 0.4f);  
int result = (int) ('C' - 5.0 / 2 + 3.5f + 0.4f);  
int result = (int) ('C' - 2.5 + 3.5f + 0.4f);  
int result = (int) (67 - 2.5 + 3.5f + 0.4f);  
int result = (int)(64.5f + 3.5f + 0.4f);  
int result = (int)(68.4f);  
int result = 68;
```

## Ejercicio 9

Indica cual será la salida por consola de las siguientes expresiones...

```
Console.WriteLine(x ?? 'C' );  
Console.WriteLine(x ?? y ?? 'C');
```

Estando definidas **x** e **y** de la siguientes formas:

1. **char? x = 'A', y = 'B';**

```
Console.WriteLine(x ?? 'C' );  
x='A';  
'A'!=null;  
// A  
  
Console.WriteLine(x ?? y ?? 'C');  
y='B';  
'B'!=null;  
//Se evalua a B, pero sigue resolviendo la expresión  
x='A';  
'A'!=null;  
// A
```

2. **char? x = null, y = 'B';**

```
Console.WriteLine(x ?? 'C' );  
x=null;  
null==null;  
// C  
  
Console.WriteLine(x ?? y ?? 'C');  
y='B';  
'B'!=null;  
//Se evalua a B, pero sigue resolviendo la expresión  
x=null;  
null=null;  
//B
```

3. **char? x = null, y = null;**



```
Console.WriteLine(x ?? 'C' );  
x=null;  
null=null;  
// C  
  
Console.WriteLine(x ?? y ?? 'C');  
y=null;  
null=null;  
//Se evalua a C, pero sigue resolviendo la expresión  
x=null;  
null=null;  
//C
```

## Ejercicio 10

Sea `x` una variable entera que almacena el valor **10**.

¿Qué almacenará después de las siguientes sentencias?

a) `y = (x > 9 ? ++x : --x);`

```
y = (10 > 9 ? ++x : --x);  
y = (T ? ++x : --x);  
y = (T ? 11 : --x);  
// y = 11    x = 11
```

`x = 11`

b) `y = (x > 9 ? X++ : x--);`

```
y = (10 > 9 ? x++ : x--);  
y = (T ? x++ : x--);  
y = (T ? 10 : x--);  
// y = 10    x = 11
```

`x = 11`

¿Y si almacena el valor 8?

c) `y = (x > 9 ? ++x : --x);`

```
y = (8 > 9 ? ++x : --x);  
y = (F ? ++x : --x);  
y = (F ? ++x : 7);  
// y = 7     x = 7
```

`x = 7`

d) `y = (x > 9 ? X++ : x--);`

```
y = (8 > 9 ? x++ : x--);  
y = (F ? x++ : x--);  
y = (F ? x++ : 8);  
// y = 8     x = 7
```

$$x = 7$$