

Operadores

Al igual que otros lenguajes de programación, PHP cuenta con operadores aritméticos, lógicos, de comparación y de asignación simple y combinada.

Combinando estos operadores con estructuras de control, es como vamos a lograr que el sistema realice lo que el programador le indique acorde a las indicaciones del usuario.

Operadores Aritméticos¹

```
1
2 //Operador de asignación
3 $nombre = "Nicolás";
4 echo "Asignación: $nombre <br>";
5
6 //Operador unario
7 $edad = 28;
8 $edad = -$edad;
9 echo "Edad: $edad <br>";
10
11 //Operador suma
12 $suma = 100+200;
13 echo "Suma: $suma <br>";
14
15 //Operador resta
16 $resta = 100-200;
17 echo "Resta: $resta <br>";
18
19 //Operador división
20 $división = 100/3;
21 echo "División: $división <br>";
22
23 //Operador módulo (resto de la división)
24 $resto = 10%3;
25 echo "Resto: $resto <br>";
26
27 //Operador multiplicación
28 $multiplicacion = 10*2;
29 echo "multiplicacion: $multiplicacion <br>";
```

Nombre: Nicolás
Edad: -28
Suma: 300
Resta: -100
División: 33.33333333333333
Resto: 1
multiplicacion: 20

Fig. 1

¹ <https://www.php.net/manual/es/language.operators.arithmetic.php>

Operadores lógicos y de comparación²

```

1  $nombre = "Nicolás";
2  $edad = 28;
3
4  //Operador igualdad
5  echo (($nombre == "Nicolás") ? 'Si':'No') . "<br>"; //true
6  echo (($nombre == "Nicolás Rotili") ? 'Si':'No') . "<br>"; //false
7
8  //Operador distinto
9  echo (($nombre ≠ "Nicolás") ? 'Si':'No') . "<br>"; //false
10 echo (($nombre ≠ "Nicolás Rotili") ? 'Si':'No') . "<br>"; //true
11
12 //Operador mayor
13 echo (($edad > 18) ? 'Si':'No') . "<br>"; //true
14
15 //Operador menor
16 echo (($edad < 18) ? 'Si':'No') . "<br>"; //false
17
18 //Operador mayor o igual
19 echo (($edad ≥ 28) ? 'Si':'No') . "<br>"; //true
20
21 //Operador menor o igual
22 echo (($edad ≤ 28) ? 'Si':'No') . "<br>"; //true
23
24 //Operador Y
25 echo (($edad ≥ 18 && $edad ≤ 30) ? 'Si':'No') . "<br>"; //true
26
27 //Operador O
28 echo (($edad ≤ 18 || $edad ≥ 28) ? 'Si':'No') . "<br>"; //true

```

Si
No
No
Si
Si
No
Si
Si
Si
Si

Fig. 2

Como se observa en la Fig. 2, los operadores probablemente se verán distinto en sus pantallas, esto se debe a un tipo de tipografía utilizado en VS (Cascadia Code) que permite tener activas las ligaduras. A continuación, se detallan nuevamente sin modificaciones:

Operador	Nombre	Resultado
\$a == \$b	Igual.	\$a es igual a \$b.
\$a === \$b	Idéntico.	\$a es igual a \$b y son del mismo tipo.
\$a <> \$b	Distinto	\$a es distinto de \$b.

² <https://www.php.net/manual/es/language.operators.comparison.php>

<code>\$a != \$b</code>	Distinto	\$a es distinto de \$b.
<code>\$a !== \$b</code>	No idéntico	\$a es distinto de \$b o no son del mismo tipo.
<code>\$a > \$b</code>	Mayor	\$a es mayor a \$b.
<code>\$a < \$b</code>	Menor	\$a es menor a \$b.
<code>\$a >= \$b</code>	Mayor o igual	\$a es mayor o igual a \$b.
<code>\$a <= \$b</code>	Menor o igual	\$a es menor o igual a \$b.
<code>&&</code>	And	\$a es verdadero y \$b es verdadero
<code> </code>	Or	\$a es verdadero o \$b es verdadero

Operadores de asignación combinada³

```

1  $numero = 10;
2
3  // Operador de incremento
4  $numero++;
5  echo "numero: $numero <br>";
6
7  // Operador de decremento
8  $numero--;
9  echo "numero: $numero <br>";
10
11 // Operador de incremento en 2
12 $numero += 2;
13 echo "numero: $numero <br>";
14
15 //Operador de decremento en 3
16 $numero -= 3;
17 echo "numero: $numero <br>";
18
19 // Operador de concatenación
20 $nombre = "Nicolás";
21 $apellido = "Rotili";
22 $nombreCompleto = $nombre . " " . $apellido;
23

```

```

número: 11
número: 10
número: 12
número: 9

```

³ <https://www.php.net/manual/es/language.operators.increment.php>

Estructuras de control

Tal como se indicó al principio del apunte, para poder darle sentido a estos operadores, debemos combinarlos con diferentes estructuras para poder darle funcionalidad a nuestro sistema siempre y cuando se cumplan condiciones.

Las estructuras de control⁴ son sentencias que permiten controlar el código que se va ejecutando basándose en ciertos factores. Por ejemplo, cuando queremos realizar una acción si cierta variable existe, o cuando queremos recorrer los valores de un array a través de un loop.

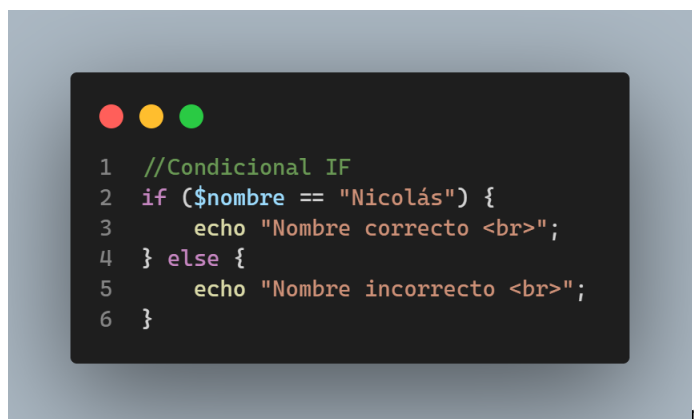
Existen muchísimas de estas estructuras, sin embargo las más utilizadas son las que se definen a continuación.

1. <code>if</code>	2. <code>else/elseif/else if</code>
3. <code>while</code>	4. <code>do while</code>
5. <code>for</code>	6. <code>foreach</code>
7. <code>break</code>	8. <code>continue</code>
9. <code>switch</code>	10. <code>match</code>
11. <code>declare</code>	12. <code>return</code>
13. <code>include/include_once</code>	14. <code>require/require_once</code>

En esta oportunidad veremos el desarrollo y funcionamiento de las estructuras de control de flujo condicional.

Condicional IF⁵

Al igual que en otros lenguajes, el operador `if` se comporta validando una operación y en caso que no la cumpla, realizar la operación que se le indique utilizando la estructura `IF`, `ELSE`, `ELSE IF`, o `ELSEIF`.



```

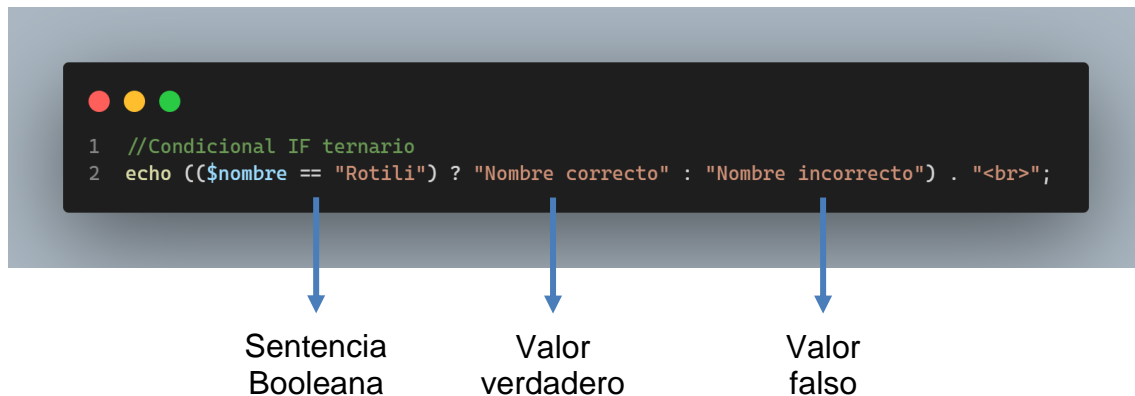
1 //Condicional IF
2 if ($nombre == "Nicolás") {
3     echo "Nombre correcto <br>";
4 } else {
5     echo "Nombre incorrecto <br>";
6 }
  
```

En este caso, utilizamos el operador "igual" para realizar la comprobación si la variable `$nombre` contiene la palabra "Nicolás". Esta comprobación, podría darnos el valor de `true` como también `false` y acorde al

⁵ <https://www.php.net/manual/es/control-structures.if.php>

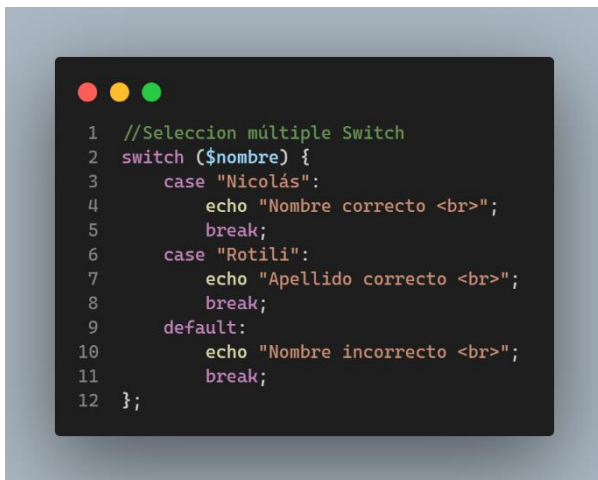
resultado será la instrucción que ejecute.

Operador ternario



Este operador, es idéntico al *condicional IF* solo que se escribe de una forma abreviada. Generalmente esta manera se utiliza al mezclarlo en HTML para evitar tener un código complejo de entender.

En ocasiones, necesitamos realizar varias comprobaciones lo cuál podríamos hacer con IF anidados pero resultaría tedioso leerlo y comprenderlo cuando se realizan muchas validaciones. PHP nos da la posibilidad de utilizar la sentencia `switch`⁶ que se comporta como varios if en una sola sentencia.




```

1 //Selección múltiple Switch
2 switch ($nombre) {
3     case "Nicolás":
4         echo "Nombre correcto <br>";
5         break;
6     case "Rotili":
7         echo "Apellido correcto <br>";
8         break;
9     default:
10        echo "Nombre incorrecto <br>";
11        break;
12 };

```

Fig. 4



```

1 if ($nombre == "Nicolás") {
2     echo "Nombre correcto <br>";
3 } elseif ($nombre == "Rotili") {
4     echo "Apellido correcto <br>";
5 } else {
6     echo "Nombre incorrecto <br>";
7 }

```

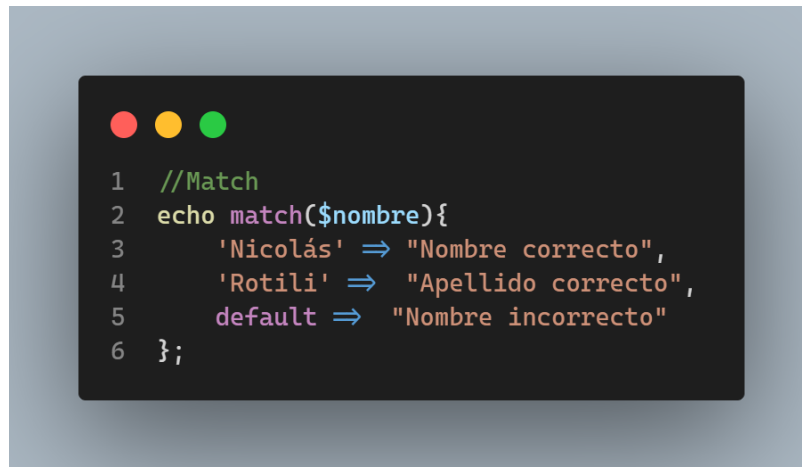
Fig. 5

Si observamos la comparación entre la Fig. 4 y 5, podemos detectar que la sentencia IF es mucha más corta, sin embargo cuando se trata de muchísimas comprobaciones, la sentencia switch es mucho más rápido de ejecutar ya que ejecuta menos líneas porque solo busca su match y luego choca con un `break`⁷ finalizando la iteración. Si se omite break, se ejecutarán todos los casos restantes cuando encuentra uno que cumpla la condición.

⁶ <https://www.php.net/manual/es/control-structures.switch.php>

⁷ <https://www.php.net/manual/es/control-structures.break.php>

A partir de su versión 8, PHP simplificó la sentencia switch en una denominada `match`⁸, sin embargo poseen una diferencia a tener en cuenta y es que, switch comprueba las coincidencias con el operador igual (==) mientras que match lo hace con el de identidad (===).



```
1 //Match
2 echo match($nombre){
3     'Nicolás' => "Nombre correcto",
4     'Rotili' => "Apellido correcto",
5     default => "Nombre incorrecto"
6 };
```

Fig. 6

⁸ <https://www.php.net/manual/es/control-structures.match.php>