

Estructuras de control - Flujo iterativo

Tal como vimos en el apunte anterior, existen muchos tipos de estructuras de control, en esta oportunidad se desarrollarán los del tipo iterativos, es decir que se ejecutan repetidamente hasta que se cumple una condición o que la detengamos con alguna sentencia.

Bucle WHILE¹

Esta estructura de repetición es una de las más sencillas en el lenguaje. Se ejecutan las sentencias dentro del while siempre y cuando la condición que se le indique sea verdadera. El valor de la condición se comprueba cada vez que inicia el bucle y la ejecución continuará hasta que deje de ser verdadera la comprobación.



En el ejemplo de la Fig. 1, inicializamos la variable `$i` en el valor de 1 para que cumpla la condición de que sea menor o igual a 10.

Como el valor de la comprobación es true, mostrará en pantalla el valor de la variable y posteriormente, utilizando el operador de asignación combinada, incrementamos ese valor en 1.

Esta operación se ejecutará hasta que el valor de `$i` sea 10.

Los dos ejemplos de la imagen están escritos de forma

distinta pero ambos cumplen la misma función.

Bucle DO - WHILE²



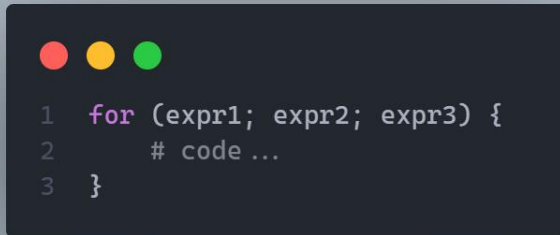
Este tipo de iteración es muy similar al del bucle while, excepto que la comprobación de condición se realiza al final de la ejecución. Por consecuencia, este tipo de estructura se va a ejecutar al menos 1 vez aunque la comprobación sea falsa (fig. 2).

¹ <https://www.php.net/manual/es/control-structures.while.php>

² <https://www.php.net/manual/es/control-structures.do.while.php>

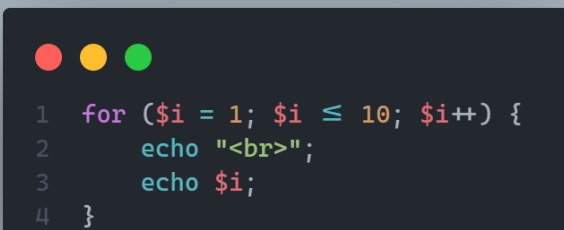
Bucle FOR³

La estructura FOR es la más compleja de PHP.



```
1  for (expr1; expr2; expr3) {  
2      # code ...  
3  }
```

Este bucle espera 3 expresiones para poder ejecutarse. La primera es ejecutada sí o sí al comenzar la estructura, la segunda es la condición (debe ser true) que debe cumplir para que corra el código y la última se ejecutará siempre y cuando la expresión 2 sea verdadera



```
1  for ($i = 1; $i <= 10; $i++) {  
2      echo "<br>";  
3      echo $i;  
4  }
```

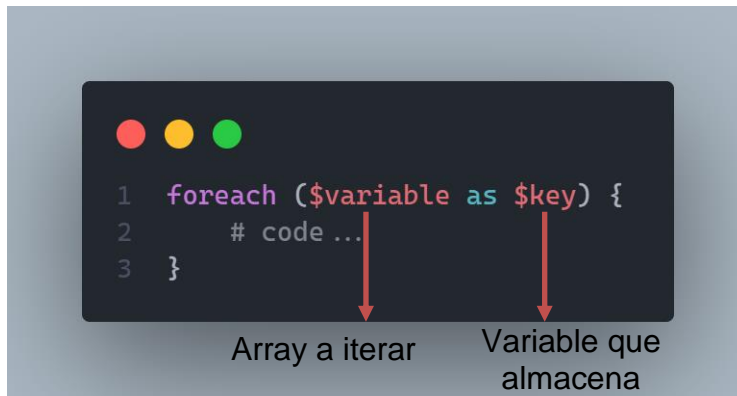
En el ejemplo de la fig. 4, iniciamos una variable \$i en 1, comprobamos que \$i sea menor o igual a 10 y si es true, la incrementamos en 1. Como la validación fue verdadera, ejecutamos el código que está dentro del for.

Constructor FOREACH

Si recordamos cuando trabajamos con arrays (comunes o asociativos), mostrar los valores resultaba complicado, más aún cuando se trata de muchos registros. Si los hiciéramos de forma manual, estaríamos no solo escribiendo muchas líneas, sino que el tratamiento del array sería mucho más lento.

Por ello, tenemos al igual que en varios lenguajes más, la estructura foreach que nos va a permitir iterar de forma sencilla dentro de un array u objeto. Si intentáramos utilizar otro tipo de variable, nos devolvería error.

³ <https://www.php.net/manual/es/control-structures.for.php>



Esta estructura nos pide dos parámetros, el primero es el array a iterar y el segundo, es donde nos almacenará el registro temporal que esté recorriendo en ese momento el array.

A modo de ejemplo, tenemos un array con datos de alumnos y lo recorreremos con un foreach.

```
1 $alumnos = [  
2   [  
3     "nombre" => "Juan",  
4     "edad" => 20,  
5     "notas" => [  
6       "mate" => 10,  
7       "fisica" => 8,  
8       "ingles" => 7  
9     ]  
10  ],  
11  [  
12    "nombre" => "Pedro",  
13    "edad" => 20,  
14    "notas" => [  
15      "mate" => 10,  
16      "fisica" => 8,  
17      "ingles" => 7  
18    ]  
19  ],  
20 ];
```

```
1 //Foreach Alumnos  
2 foreach ($alumnos as $alumno) {  
3   echo "Nombre: ".$alumno['nombre']." <br>";  
4   echo "Edad: ".$alumno['edad']." <br>";  
5   echo "Notas: <br>";  
6   echo "Mate: ".$alumno['notas']['mate']." <br>";  
7   echo "Fisica: ".$alumno['notas']['fisica']." <br>";  
8   echo "Ingles: ".$alumno['notas']['ingles']." <br>";  
9   echo "<hr>";  
10 }
```

Como se puede observar en estas dos imágenes, es mucho más sencillo recorrer un array utilizando las estructuras adecuadas que invocar cada registro a través de su índice.

Esta estructura tiene la particularidad de que más allá de poder ver lo contenido en una variable, también podemos trabajar con sus índices y de esta manera, simplificar las formas de mostrar los datos contenidos.

En la figura a continuación, realizamos el mismo trabajo hecho en la fig. 6 pero en esta ocasión haciendo uso de los índices.

```
1 foreach ($alumnos as $key => $alumno) {  
2   echo "Nombre: ".$alumno['nombre']." <br>";  
3   echo "Edad: ".$alumno['edad']." <br>";  
4   echo "Notas: <br>";  
5   foreach ($alumno['notas'] as $key => $nota) {  
6     echo $key.": ".$nota." <br>";  
7   }  
8   echo "<hr>";  
9 }
```

Estructura **BREAK**⁴

Si hacemos memoria, cuando vimos condicionales utilizamos break en la estructura switch.

Break; es una estructura que se puede utilizar en for, foreach, while, do-while o switch para finalizar la ejecución de la iteración que se está llevando a cabo.

De esta forma, uno puede, al cumplir alguna condición determinada salir de la iteración sin necesidad de que termine de ejecutarse. Tal es el caso de switch que, una vez que encuentra su coincidencia, ejecuta las sentencias que le corresponde y sale de la iteración.

Esta estructura, tiene la particularidad de que se la puede acompañar de algún valor numérico luego de invocarla (su valor por defecto es 1), por ejemplo break 2; el cual indica cuantas estructuras anidadas debe finalizar.



```
1 $i = 0;
2 while (++$i) {
3     switch ($i) {
4         case 5:
5             echo "En 5, salgo del switch pero
6                 continúo ejecutando el while.<br />";
7             break 1;
8         case 10:
9             echo "En 10; salgo del switch y del while.<br />";
10            break 2;
11        default:
12            break;
13    }
14 }
```

Fig. 7

Estructura **CONTINUE**⁵



```
1 for ($i = 0; $i <= 10; ++$i) {
2     if ($i % 2 != 0) {
3         continue;
4     }
5     echo "$i\n";
6 }
7
```

Esta estructura se utiliza en los bucles para saltar el resto de la iteración y continuar con la próxima repetición. Al igual que break, se le puede adicionar un argumento opcional para indicar cuantos niveles de bucles debe omitir.

Si observamos el código de la Fig. 8, notaremos la particularidad de que solo nos mostrará los números pares entre 0 y 10.

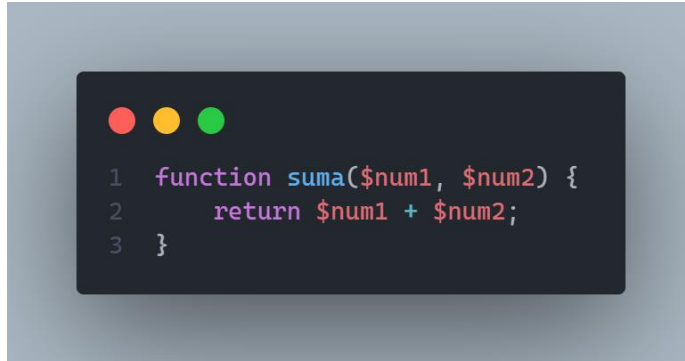
⁴ <https://www.php.net/manual/es/control-structures.break.php>

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

Constructor RETURN⁶

Return se utiliza para devolverle el control al módulo que lo invoca. Generalmente este tipo de constructor es muy utilizado cuando se generan funciones para que, una vez finalizada su ejecución nos devuelva el control, ya sea con un valor o simplemente para indicarnos su finalización.

También suele utilizarse en archivos externos (al final del documento) que son invocados mediante include o require.



```

1 function suma($num1, $num2) {
2     return $num1 + $num2;
3 }

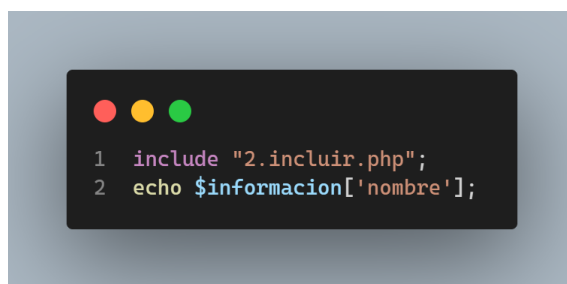
```

En el ejemplo de la Fig. 8, utilizamos esta estructura (línea 2) para devolverle el control al módulo que invoca la función y, a su vez, le pasamos el resultado de la suma de lo contenido en las variables \$num1 y \$num2.

Estructura INCLUDE⁷ / INCLUDE ONCE⁸

Las estructuras mencionadas se utilizan para incluir un archivo en otro y ejecutarlo. Es utilizada en muchas ocasiones cuando tenemos archivos que contienen scripts que podemos reutilizar en muchas páginas, por ejemplo, un archivo de funciones o con la conexión a una base de datos.

La diferencia entre INCLUDE e INCLUDE_ONCE es que el segundo solo incluye el archivo una sola vez, si se lo vuelve a invocar en la misma página y este ya fue incluido, no lo vuelve a hacer, evitando así redundancias de funciones o variables.

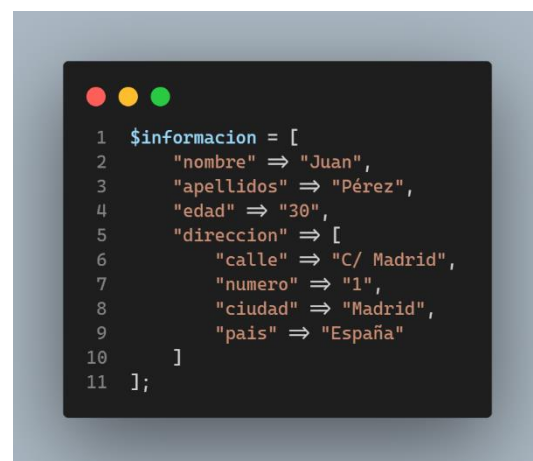


```

1 include "2.incluir.php";
2 echo $informacion['nombre'];

```

1.tipos.php



```

1 $informacion = [
2     "nombre" => "Juan",
3     "apellidos" => "Pérez",
4     "edad" => "30",
5     "direccion" => [
6         "calle" => "C/ Madrid",
7         "numero" => "1",
8         "ciudad" => "Madrid",
9         "pais" => "España"
10    ]
11 ];

```

2.incluir.php

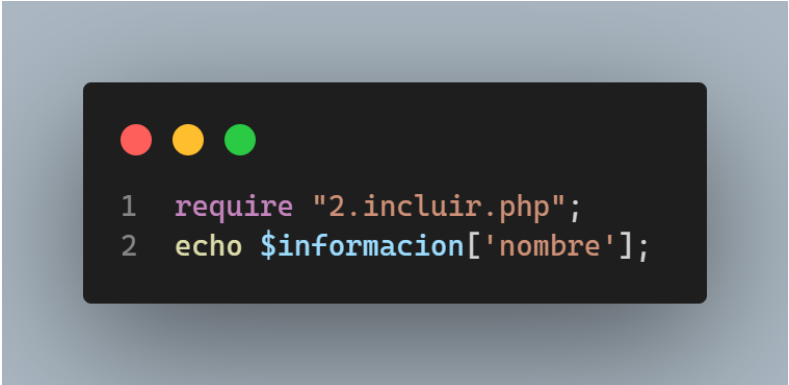
⁶ <https://www.php.net/manual/es/function.return.php>

⁷ <https://www.php.net/manual/es/function.include.php>

⁸ <https://www.php.net/manual/es/function.include-once.php>

Estructura REQUIRE⁹ / REQUIRE ONCE¹⁰

Este tipo se comporta de manera similar a la estructura anterior. Se diferencia en que, si hay algún problema al cargar el archivo, se detiene la ejecución de todos los scripts, aún así del archivo que lo invoca. Con `include` o `include_once`, solo nos dará un error de advertencia pero seguirá ejecutándose la página.



```
1 require "2.incluir.php";  
2 echo $informacion['nombre'];
```

⁹ <https://www.php.net/manual/es/function.require.php>

¹⁰ <https://www.php.net/manual/es/function.require-once.php>