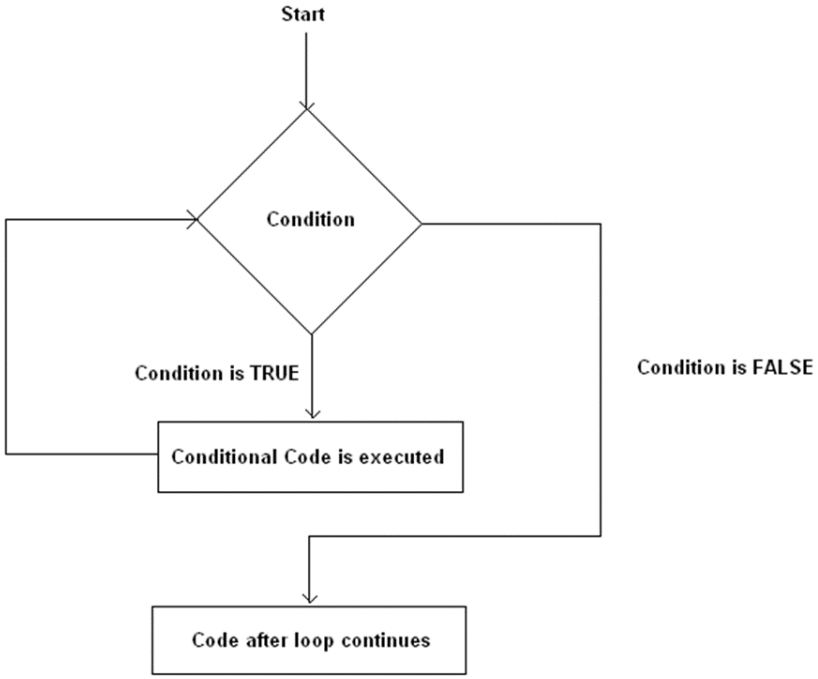
“LOS BUCLES FOR, FOREACH, WHILE EN PHP "

Bucles

Los bucles en PHP son útiles cuando desea ejecutar un fragmento de código repetidamente hasta que una condición se evalúe como falsa. Por lo tanto, el código se ejecuta repetidamente siempre que una condición se evalúe como verdadera y, tan pronto como la condición se evalúe como falsa, la secuencia de comandos continúa ejecutando el código después del bucle.

El siguiente diagrama de flujo explica cómo funcionan los bucles en PHP.



Como puede ver en la captura de pantalla anterior, un bucle contiene una condición. Si la condición se evalúa como verdadera, se ejecuta el código condicional. Después de la ejecución del código condicional, el control vuelve a la condición de bucle y el flujo continúa hasta que la condición se evalúa como falsa.

En esta sección, veremos los diferentes tipos de bucles admitidos en PHP.

Bucle While

El bucle while se usa cuando se desea ejecutar un fragmento de código repetidamente hasta que la condición while se evalúa como falsa.

Puedes definirlo como se muestra en el siguiente pseudo código.

```
while (expression)
{
  // code to execute as long as expression evaluates to TRUE
}
```

Veamos un ejemplo del mundo real para comprender cómo funciona el bucle while en PHP.

```
<?php
$max = 0;
echo $i = 0;
echo ",";
echo $j = 1;
echo ",";
$result=0;
while ($max < 10 )
{
   $result = $i + $j;
   $i = $j;
   $j = $result;
   $max = $max + 1;
   echo $result;
```

```
    echo ",";

}

?>
```

Si está familiarizado con la serie de Fibonacci, puede reconocer lo que hace el programa anterior: genera la serie de Fibonacci para los primeros diez números. El bucle while se usa generalmente cuando no se sabe la cantidad de iteraciones que se llevarán a cabo en un bucle.

Bucle Do-While

El bucle do-while es muy similar al bucle while, con la única diferencia de que la condición while se verifica al final de la primera iteración. Por lo tanto, podemos garantizar que el código de bucle se ejecute al menos una vez, independientemente del resultado de la expresión while.

Echemos un vistazo a la sintaxis del bucle do-while.

```
do

{

  // code to execute

} while (expression);
```

Vayamos a un mundo real para comprender posibles casos de uso en los que puede usar el bucle do-while.

```
<?php

$handle = fopen("file.txt", "r");

if ($handle)

{

    do

    {

        $line = fgets($handle);
```

```
        // process the line content

    } while($line !== false);

}

fclose($handle);

?>
```

En el ejemplo anterior, estamos intentando leer un archivo línea por línea. En primer lugar, hemos abierto un archivo para su lectura. En nuestro caso, no estamos seguros de si el archivo contiene algún contenido. Por lo tanto, debemos ejecutar la función fgets al menos una vez para verificar si un archivo contiene algún contenido. Así que podemos usar el bucle do-while aquí. do-while evalúa la condición después de la primera iteración del bucle.

Blucle For

Generalmente, el bucle for se usa para ejecutar un fragmento de código por un número específico de veces. En otras palabras, si ya conoce la cantidad de veces que desea ejecutar un bloque de código, es el bucle for que es la mejor opción.

Echemos un vistazo a la sintaxis del bucle for.

```
for (expr1; expr2; expr3)

{

  // code to execute

}
```

La expresión expr1 se usa para inicializar variables, y siempre se ejecuta. La expresión expr2 también se ejecuta al comienzo de un bucle, y si se evalúa como verdadera, se ejecuta el código de bucle. Después de la ejecución del código de bucle, se ejecuta el expr3. Generalmente, el expr3 se usa para alterar el valor de una variable que se usa en la expresión expr2.

Veamos el siguiente ejemplo para ver cómo funciona.

```
<?php
```

```
for ($i=1; $i<=10; ++$i)

{

  echo sprintf("The square of %d is %d.</br>", $i, $i*$i);

}

?>
```

El programa anterior produce el cuadrado de los primeros diez números. Inicializa $ i a 1, se repite mientras $ i sea menor o igual a 10, y agrega 1 a $ i en cada iteración.

For Each

El bucle foreach se utiliza para iterar sobre variables de matriz. Si tiene una variable de matriz y desea recorrer cada elemento de esa matriz, el bucle foreach es la mejor opción.

Echemos un vistazo a un par de ejemplos.

```
<?php

$fruits = array('apple', 'banana', 'orange', 'grapes');

foreach ($fruits as $fruit)

{

  echo $fruit;

  echo "<br/>";

}

$employee = array('name' => 'John Smith', 'age' => 30, 'profession' => 'Software Engineer');

foreach ($employee as $key => $value)

{

  echo sprintf("%s: %s</br>", $key, $value);

  echo "<br/>";
```

```
}

?>
```

Si desea acceder a valores de matriz, puede usar la primera versión del bucle foreach como se muestra en el ejemplo anterior. Por otro lado, si desea acceder tanto a una clave como a un valor, puede hacerlo como se muestra en el ejemplo de $ employee anterior.

En **PHP**, los **ciclos** (o bucles) permiten ejecutar un bloque de código repetidamente mientras se cumpla una condición o hasta recorrer una colección de datos.

Aquí tienes un resumen de los **tipos de ciclos en PHP** con ejemplos completos y funcionales.

1. while

Ejecuta el bloque **mientras** la condición sea verdadera.

Php

```php
<?php
$contador = 1;

while ($contador <= 5) {
    echo "Número: $contador<br>";
    $contador++;
}
?>
```

Uso típico: Cuando no sabemos cuántas iteraciones habrá y la condición se evalúa antes de entrar al ciclo.

2. do...while

Ejecuta el bloque **al menos una vez** y luego repite mientras la condición sea verdadera.

Php

```php
<?php
$contador = 1;

do {
    echo "Número: $contador<br>";
    $contador++;
} while ($contador <= 5);
?>
```

Uso típico: Cuando queremos que el código se ejecute al menos una vez, sin importar la condición inicial.

3. for

Se usa cuando conocemos el número exacto de iteraciones.

Php

```php
<?php
for ($i = 1; $i <= 5; $i++) {
    echo "Número: $i<br>";
}
?>
```

Uso típico: Recorridos con contador definido.

4. foreach

Especial para recorrer **arrays** o **objetos**.

Php

```php
<?php
$frutas = ["Manzana", "Banana", "Cereza"];

foreach ($frutas as $fruta) {
    echo "Fruta: $fruta<br>";
}
?>
```

Con clave y valor:

Php

```php
<?php
$edades = ["Juan" => 25, "Ana" => 30, "Luis" => 28];

foreach ($edades as $nombre => $edad) {
    echo "$nombre tiene $edad años<br>";
}
?>
```

Uso típico: Recorrer listas o colecciones de datos.

5. Control de ciclos

- **break** → Sale del ciclo inmediatamente.
- **continue** → Salta a la siguiente iteración.

Php

```php
<?php
for ($i = 1; $i <= 5; $i++) {
    if ($i == 3) {
        continue; // Salta el número 3
    }
```

```
    if ($i == 5) {
        break; // Detiene el ciclo en 5
    }
    echo "Número: $i<br>";
}
?>
```