Unidad 4. Condicionales y Bucles

Las sentencias condicionales permiten bifurcar el flujo del programa en función del cumplimiento o no de una o varias condiciones.

Sentencia `if`

La sentencia if permite la ejecución de una serie de instrucciones dependiendo del resultado de evaluar una expresión lógica[1]. Se podría traducir al español como "si se cumple esta condición haz esto y si no, haz esto otro".

El formato del `if` es el siguiente:

```
if (condición) {

  sentencias a ejecutar cuando la condición es cierta

} else {

  Sentecias a ejecutar cuando la condición es falsa.

}
```

A continuación se muestra un ejemplo completo en PHP:

```php
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
      $a = 8;
      $b = 3;
      if ($a < $b) {
        echo "a es menor que b";
      } else {
        echo "a no es menor que b";
      }
```

[1]El resultado de evaluar una expresión lógica siempre es verdadero o falso.

```
    ?>
  </body>
</html>
```

En este ejemplo, la condición no es verdadera por lo que se ejecuta la parte de código correspondiente al `else`.

En PHP también existe la posibilidad de utilizar una condición con la pareja de símbolor `?` y `:` al estilo de C. El formato es el siguiente:

```
(condición) ? expresión1 : expresión2
```

El funcionamiento es el siguiente: se evalúa la condición; si la condición es verdadera, el resultado que se devuelve es `expresión1` y si la condición es falsa, se devuelve `expresión2`.

A continuación se muestra un programa que ilustra este tipo de sentencia.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      $x = (20 > 6) ? 11 : 22;
      echo $x;
    ?>
  </body>
</html>
```

La expresión `(20 > 6)` es verdadera, por tanto en la variable `$x` se almacena el número `11` que es lo que se muestra por pantalla. Prueba a cambiar la condición para que sea falsa para comprobar que, esta vez, se muestra el `22`.

Operadores de comparación

En el ejemplo de la sección anterior se utiliza el operador `<` en la comparación `if ($a < $b)` para saber si el valor de la variable `$a` es menor que el de la variable `$b`. Existen más operadores de comparación, a continuación se muestran todos ellos en una tabla:

Operador	Nombre	Ejemplo	Devuelve verdadero cuando…
==	Igual	$a == $b	$a es igual $b (aunque sean de diferente tipo)
===	Igual	$a === $b	$a es igual $b (y además son del mismo tipo)
!=	Distinto	$a != $b	$a es distinto $b
<	Menor que	$a < $b	$a es menor que $b
>	Mayor que	$a > $b	$a es mayor que $b
<=	Menor o igual	$a <= $b	$a es menor o igual que $b
>=	Mayor o igual	$a >= $b	$a es mayor o igual que $b
<=>	Nave espacial	$a <=> $b	-1 si $a es menor, 0 si son iguales y 1 si $b es menor

En el siguiente programa se muestra claramente la diferencia entre `==` y `===`.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      $a = 5;
      $b = "5";
      echo ($a == 5)?"verdadero":"falso", "<br>";
      echo ($a === 5)?"verdadero":"falso", "<br>";
      echo ($b == 5)?"verdadero":"falso", "<br>";
      echo ($b === 5)?"verdadero":"falso", "<br>";
    ?>
  </body>
</html>
```

El resultado del programa sería el siguiente:

```
verdadero
verdadero
verdadero
falso
```

Operadores lógicos

Las comparaciones se pueden combinar mediante los operadores lógicos. Por ejemplo, si queremos saber si la variable `$a` es mayor que `b$` y además menor que `$c`, escribiríamos `if (($a > $b) && ($a < $c))`. La siguiente tabla muestra los operadores lógicos de PHP:

Operador	Nombre	Ejemplo	Devuelve verdadero cuando…
&&	Y	(7>2) && (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
and	Y	(7>2) and (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
||	O	(7>2) || (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
or	O	(7>2) or (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
!	No	! (7>2)	Niega el valor de la expresión.

A continuación se muestra el uso de estos operadores lógicos en un programa PHP:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
      $a = 8;
      $b = 3;
      $c = 3;
      echo ($a == $b) && ($c > $b),"<br>";
      echo ($a == $b) || ($b == $c),"<br>";
      echo !($b <= $c),"<br>";
    ?>
  </body>
</html>
```

Observa que cuando el resultado de evaluar la expresión lógica es verdadero, se muestra un `1` por pantalla.

Sentencia `switch`

En ocasiones es necesario comparar el valor de una variable con una serie de valores concretos. Es algo parecido (aunque no exactamente igual) a utilizar varios `if` consecutivos.

El formato del `switch` es el siguiente:

```
switch(variable)

  { case valor1:
    sentencias
    break;

  case valor2:
    sentencias
    break;
    .
    .
    .

  default:
    sentencias
}
```

A continuación se muestra un ejemplo completo en PHP:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title></title>
  </head>
  <body>
    <?php
      $posicion = "arriba";

      switch($posicion) {
        case "arriba":  // Bloque 1
          echo "La variable contiene";
          echo " el valor arriba";
          break;
        case "abajo": // Bloque 2
          echo "La variable contiene";
          echo " el valor abajo";
          break;
        default:  // Bloque 3
          echo "La variable contiene otro valor";
          echo " distinto de arriba y abajo";
      }
    ?>
  </body>
</html>
```

Prueba a cambiar el valor de la variable `$posicion` del ejemplo anterior y observa cómo se ejecuta el bloque de sentencias correspondiente.

Bucles

Los bucles se utilizan para repetir un conjunto de sentencias.

Bucle `for`

Se suele utilizar cuando se conoce previamente el número exacto de iteraciones que se van a realizar. La sintaxis es la siguiente:

```
for (expresion1 ; expresion2 ; expresion3) {

  sentencias

}
```

Justo al principio se ejecuta `expresion1`, normalmente se usa para inicializar una variable. El bucle se repite mientras se cumpla `expresion2`. En cada iteracción del bucle se ejecuta `expresion3`, que suele ser el incremento o decremento de una variable. A continuación se muestra un ejemplo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
      for($i = 0 ; $i < 10 ; $i++) {
        echo "El valor de i es ", $i,"<br>";
      }
    ?>
  </body>
</html>
```

Bucle `while`

El bucle `while` se utiliza para repetir un conjunto de sentencias siempre que se cumpla una determinada condición. Es importante reseñar que la condición se comprueba al comienzo del bucle, por lo que se podría dar el caso de que dicho bucle no se ejecutase nunca. La sintaxis es la siguiente:

```
while (expresion) {

  sentencias

}
```

Las sentencias se ejecutan una y otra vez mientras `expresion` sea verdadera. El siguiente ejemplo produce la misma salida que el ejemplo anterior, muestra cómo cambian los valores de `$i` del 0 al 9.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
      $i = 0;
      while( $i < 10 ) {
        echo "El valor de i es ", $i,"<br>"; $i++;
      }
    ?>
  </body>
</html>
```

Bucle `do-while`

El bucle `do-while` funciona de la misma manera que el bucle `while`, con la salvedad de que expresion se evalúa al final de la iteracción. Las sentencias que encierran el bucle `do-while`, por tanto, se ejecutan como mínimo una vez. La sintaxis es la siguiente:

```
do {

  sentencias

} while (expresion)
```

El siguiente ejemplo es el equivalente `do-while` a los dos ejemplos anteriores.

```php
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
    $i = 0;
    do {
      echo "El valor de i es ", $i,"<br>"; $i++;
    } while( $i < 10 )
    ?>
  </body>
</html>
```

Carga reiterada de una página. Adivina el número.

Los bucles `for`, `while` y `do-while` permiten repetir un bloque de sentencias en una carga de página. No obstante, cada vez que un programa necesita pedir información al usuario, es necesario presentar un formulario por pantalla, el usuario rellena ese formulario y los datos son enviados a una página que los procesa que puede ser la misma página que envía los datos. Si se vuelven a necesitar datos del usuario, se repite el proceso. Podemos tener así un bucle, es decir, una repetición de sentencias, que se produce por la carga sucesiva de la misma página y no por la utilización de `for`, `while` o `do-while`

Lo entenderemos mejor con un ejemplo. Vamos a realizar un pequeño juego al que llamaremos “Adivina el número”. El usuario debe adivinar un número entre 0 y 100; el programa irá guiando al usuario diciendo si el número introducido es mayor o menor que el que está pensando el ordenador. Tendremos en primer lugar una página llamada `index.php` que se utiliza únicamente para inicializar las variables:

```php
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    Adivina el número que estoy pensando.
    <form action="adivina.php" method="post">
      <input type="hidden" name="numeroIntroducido" value="555">
      <input type="hidden" name="oportunidades" value="5">
      <input type="submit" value="Continuar">
    </form>
  </body>
</html>
```

Fíjate que se inicializa `oportunidades` a 5; ese será el número de oportunidades que tendrá el usuario para adivinar el número. La variable `numeroIntroducido` se inicializa de forma arbitraria a 555 simplemente para

indicarle a la página principal del juego cuándo se ejecuta por primera vez. Este número se puede cambiar pero debe quedar siempre fuera del intervalo [0 - 100] para que no se confunda con alguno de los números introducidos por el usuario.

A continuación se muestra la página `adivina.php` que constituye el cuerpo principal del programa:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
  </head>
  <body>
    <?php
      $oportunidades = $_POST['oportunidades'];
      $numeroIntroducido = $_POST['numeroIntroducido'];

      $numeroSecreto = 24;

      if ($numeroIntroducido == $numeroSecreto) {
        echo "Enhorabuena, has acertado el número.";
      } else {
        if ($oportunidades == 0) {
          echo "Lo siento, has agotado todas tus oportunidades. Has perdido";
        } else {
          if ($numeroIntroducido!=555) {
            if ($numeroSecreto > $numeroIntroducido)
              echo "El número que estoy pensando es mayor que el número que has introducid\
o.<br>";
            else
              echo "El número que estoy pensando es menor que el número que has introducid\
o.<br>";
          }
        ?>
        Te quedan <?= $oportunidades-- ?> oportunidades para adivinar el número.<br>
        Introduce un número del 0 al 100
        <form action="adivina.php" method="post">
          <input type="text" name="numeroIntroducido">
          <input type="hidden" name="oportunidades" value="<?= $oportunidades ?>">
          <input type="submit" value="Continuar">
        </form>
        <?php
        }
      }
    ?>
  </body>
</html>
```