4. Bucles

Los bucles se utilizan para repetir un conjunto de sentencias.

4.1 El 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:

```php
<!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>

```

El valor de i es 0
El valor de i es 1
El valor de i es 2
El valor de i es 3
El valor de i es 4
El valor de i es 5
El valor de i es 6
El valor de i es 7
El valor de i es 8
El valor de i es 9

4.2 El 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>
```

4.3 El 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>
```

4.4 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:

```html
<!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="500">
        <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 500 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:

```php
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<?php
$oportunidades = $_POST['oportunidades'];
$numeroIntroducido = $_POST['numeroIntroducido'];
$numeroSecreto = 36;

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 ($numeroIntroducido != 500) {
            if ($numeroSecreto > $numeroIntroducido)
                echo "El número que estoy pensando es mayor que el número que has introducido.<br>";
            else
                echo "El número que estoy pensando es menor que el número que has introducido.<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>
```

Ejercicio 4.1

Muestra los números del 400 al 160, contando de 20 en 20 utilizando un bucle for.

Ejercicio 4.2

Realiza el control de acceso a una caja fuerte. La combinación será un número de 4 cifras. El programa nos pedirá la combinación para abrirla. Si no acertamos, se nos mostrará el mensaje “Lo siento, esa no es la combinación” y si acertamos se nos dirá “La caja fuerte se ha abierto satisfactoriamente”. Tendremos cuatro oportunidades para abrir la caja fuerte.

Ejercicio 4.3

Muestra la tabla de multiplicar de un número introducido por teclado.

Ejercicio 4.4

Realiza un programa que nos diga cuántos dígitos tiene un número introducido por teclado.

Ejercicio 4.5

Realiza un programa que nos diga cuántos dígitos tiene un número introducido por teclado.

Ejercicio 4.6

Escribe un programa que lea una lista de diez números y determine cuántos son positivos, y cuántos son negativos.

Ejercicio 4.7

Escribe un programa que pida una base y un exponente (entero positivo) y que calcule la potencia.

Ejercicio 4.8

Escribe un programa que diga si un número introducido por teclado es o no primo. Un número primo es aquel que sólo es divisible entre él mismo y la unidad.

Ejercicio 4.9

Realiza un programa que pida un número por teclado y que luego muestre ese número al revés.

Ejercicio 4.10

Realiza una web que pida un número. Una vez elegido el número comenzará una cuenta atrás que se decrementará en una unidad cada vez que se pulse un botón “Siguiente”.

Ejercicio 4.11

Realiza un programa que vaya pidiendo números hasta que se introduzca un numero negativo y nos diga cuantos números se han introducido, la media de los impares y el mayor de los pares . El número negativo sólo se utiliza para indicar el final de la introducción de datos pero no se incluye en el cómputo.

Ejercicio 4.12

Realiza un programa que muestra las tablas de multiplicar. Mostrará sólo una a la vez. Tendrá botones de anterior y siguiente para ir a la tabla anterior y posterior. En el caso del 1 no habrá botón anterior y en el caso del 10 no habrá botón de siguiente.