El bucle for

Se trata de una nueva forma –de uso bastante habitual– que permite establecer un bucle que se repetirá mientras una variable numérica se mantenga dentro de intervalo -establecido en la sintaxis del propio bucle– indicándose, también en la propia instrucción, el criterio de modificación de esa variable en cada ejecución del bucle.

La sintaxis es la siguiente:

for (*desde* **;** *hasta* **;** *incre* **){**
.....
...instrucciones....
.....
}

El parámetro **desde** permite asignar un **valor inicial** a una variable (**$var=***num*) que hará funciones de *controladora de iteraciones*.

El parámetro **hasta** establece la condición que limita el valor máximo que puede alcanzar la variable de *control*.

El parámetro **incre** (con una sintaxis del tipo *$variable++*; *$variable--*; *++$variable* -- *$variable*; *$variable +=n* o *$variable -=n* establece los *incrementos* o *decrementos* de la *variable controladora* en cada iteración del bucle.

Las intrucciones contenidas entre **{ }** serán ejecutadas cada vez que se reitere el bucle.

Variantes del bucle for

El bucle for permite algunas variantes respecto a su forma más general. Son estas:

for (*desde* **; ;** *incre* **){**
.....
...instrucciones....
.....
}

En este caso se omite el valor del parámetro **hasta** (observa que no se omite el separador de parámetros (**;**) con lo que en realidad se está asignando a *hasta* el valor NUL.

Cuando se utiliza esta sintaxis, el bucle se repetiría de forma **indefinida** (la variable podría tomar *cero* como valor, pero, cero es distinto de NUL) salvo que -tal como puedes ver en el ejemplo- se escriba en las **instrucciones** un operador condicional con una opción de *ruptura del bucle* -el famoso **break** que ya hemos visto al estudiar la instrucción **while** y otras anteriores–.

for (; ;){
.....
...instrucciones....
.....
}
En este caso no se inserta ningún parámetro pero si se escriben los **;** delimitadores de los mismos.

Si observas el ejemplo verás que el control se realiza fuera del *for*. El valor de la *variable contador* se asigna *fuera* del bucle, los incrementos de esa variable están escritos en las líneas de

La estructura for

Aquí tienes algunos ejemplos de las diferentes variantes del bucle for.

```
<?
for ($i = 1; $i <= 10; $i++) {
     print $i."<br>";
 }
 ?>
```

ejemplo45.php

```
<?
for ($i = 1;;$i++) {
     if ($i > 10) {
          break;
     }
     print $i."<br>";
 }
?>
```

ejemplo46.php

```
<?
$i = 1;
for (;;) {
     if ($i > 10) {
          break;
     }
     print $i."<br>";
     $i++;
 }
 ?>
```

ejemplo47.php

```
<?
for ($i = 1; $i <= 10; print $i."<br>", $i++) ;
?>
```

ejemplo48.php

```
<?
for($i = 1; $i <=10;$i++):
        echo $i,"<br>";
endfor;
?>
```

ejemplo49.php

```
<? for ($i = 1; $i <= 10;$i++):?>
        <H1>Esto se repetirá 10 veces</H1>
<? endfor; ?>
```

ejemplo50.php

Como puedes observar en este último ejemplo también es aplicable aquí la sintaxis de los *dos sripts PHP*. El primero contiene las instrucciones del bucle y el segundo señala el final del mismo.

Entre ambos *scripts* se escribe el código HTML

¡Cuidado!

A la hora de programar bucles hay que evitar el riesgo de convertirlo en un bucle indefinido. Cuando esto ocurre –el error es humano– al abrir la página que lo contiene parecerá que nuestro navegador se ha quedado *colgado* aunque en realidad estará esperando a que sea atendida la **petición**.
Si llega a planteársete ese problema, tendrás que recurrir a la *socorrida* solución de pulsar ALT+CTRL+DEL para abortar la petición del navegador.

Ejercicio nº 23

Siguiendo criterios similares a los del *ejemplo nº 40* elabora un script que permita construir

instrucciones y llevan **un operador condicional** con la función **break** para permitir la salida.

for(*desd* **;** *hast* **;** *inst, incr* **)**

Esta nueva variante de **for** permite insertar instrucciones a través del tercer parámetro de la función.

Si insertamos como tercer parámetro una conjunto de instrucciones, **separadas por comas**, se ejecutarán de igual forma que si estuvieran contenidas entre **{** y **}**

En este caso, el modificador de la variable de control (*incr*) se incluye como una instrucción más –separada por una coma– de las contenidas en ese tercer parámetro de la función.

for (*desde* **;** *hasta* **;** *incre* **):**
.....
...instrucciones....
.....
endfor;

Esta sintaxis es alternativa a la primera de las descritas. Sustituye la **{** por dos puntos (**:**) y la **}** por **endfor**.

una tabla de 5 filas y 7 columnas que contengan los sucesivos números naturales desde 1 hasta 35. Utiliza bucles del tipo **for**, que igual que while y do while permiten ser anidados. Guárdalo como **ejercicio23.php**

Ejercicio nº 24

En este ejercicio –**ejercicio24.php**– trataremos de crear una tabla como la anterior, esta vez de una sola fila y seis columnas, conteniendo cada celda un **número aleatorio** comprendido entre **1 y 49** en la que habremos de evitar la posibilidad de que un número se repita dos veces (podría ser una forma de rellenar *la primitiva*).

Para ello te sugerimos que guardes en un array los valores de los números aleatorios que se van generando y que, antes de guardar cada uno de ellos, se ejecute un bucle que compruebe si entre los registrados ya existe un valor igual al obtenido. Si no existiera ese valor se guardaría el dato, en caso contrario se repetiría la extracción.