Los bucles

La necesidad de *repetir* la ejecución de instrucciones es algo habitual en el mundo de la programación.

Frente a la alternativa –poco práctica– de rescribir esas instrucciones todos los lenguajes de programación disponen de funciones que pueden ejecutar un bloque de instrucciones de forma repetitiva.

La instrucción while

Como ocurría en el caso de *if*, el parámetro *condición* permite cualquier estructura lógica, y también dispone de distintas opciones de sintaxis.

while(*condición***)**
...instrucción

Con esta sintaxis estaremos indicando que la *instrucción siguiente* (sólo **una** instrucción) ha de ejecutarse continua y repetidamente hasta que deje de cumplirse la condición establecida.

while(*condición***){**
...instrucción
.....
}

De forma similar a la utilizada en el caso de *if*, también en este caso, las llaves hacen la función de *contenedores* de las instrucciones cuya ejecución debe repetirse mientras se cumpla la condición.

while(*condición***):**
...instrucción
.....
endwhile;

También aquí se mantiene la similitud con la sintaxis del condicional *if*.
La *llave* (**{**) pueden sustituirse por (**:**) y en este caso en vez de (**}**) habría que escribir **endwhile**.

while(*condición***) : ?>**
...etiquetas HTML
.....
<? endwhile; ?>

También while permite *cerrar el script* PHP después de (**:**) o de la sintaxis alternativa (**{**) e insertar etiquetas HTML, indicando más tarde el final del bucle con **<? } ?>** o **<? endwhile; ?>**, según proceda.

Whiles anidados

La instrucción while

```php
<?
# asignemos un valor a la variable $A
$A=0;
/* establezcamos la condición menor que cinco
 e insertemos dentro de la instrucción algo que modifique
 el valor de esa variable de modo que en algun momento
 deje de cumplirse la condición;
 de no ocurrir esto, el bucle se repetiría indefinidamente
 en este ejemplo el autoincremento ++ de la variable
 hará que vaya modificandose su valor*/

while ($A<5) echo "El valor de A es: ",$A++,"<br>";
# comprobemos que este while solo ejecuta una instrucción
# la delimitada por el punto y coma anterior
print("Esto solo aparecerá una vez. While no lo incluye en su bucle");
?>
```

ejemplo38.php

```php
<?
$A=0;
/* utilicemos ahora el bucle para crear un tabla HTML
   empecemos escribiendo la etiqueta de apertura de esa tabla
   fuera del bucle (ya que esa se repite una sola vez)
   y utilicemos el bucle para escribir las celdas y sus contenidos */

print ("<table width=300 border=2>");

while ($A<=5){
      echo "<tr><td align=center>";
      print $A;
# esta instrucción es importantisima
# si no modificamos el valor de $A el bucle sería infinito
          $A++;
      print("</td></tr>");
}
# cerremos la etiqueta table
print "</table>";
?>
```

ejemplo39.php

```php
<?
# utilicemos whiles anidados para construir una tabla de
$filas=5; $columnas=3;
# insertemos la etiqueta de apertura de la tabla
print ("<table border=2 width=400 align=center>");
# un primer while rojo que utiliza la condición  filas mayor que cero
# en este caso, la variable tendrá que ir disminuyendo su valor con $filas
# para escribir las etiquetas <tr> y </tr>
# y el modificador de la variable filas
# y un segundo while (magenta) para insertar las etiquetas correspondiente
# a las celdas de cada fila
while ($filas>0):
    echo "<tr>";
    $filas--;
        while ($columnas>0):
                echo "<td>";
                print "fila: ".$filas." columna: ".$columnas;
                print ("</td>");
```

Una nueva similitud sintáctica con *if*. En el caso de while también es posible insertar un while dentro de otro while utilizando una sintaxis de este tipo:

while(*condición1*)**:**
...instrucción
while(condición2) {
...instrucción
.....
}
.....
endwhile;

En esta descripción hemos utilizado dos sintaxis distintas. Por si acaso dudaras de si es necesario o no hacerlo de esa forma, te diremos que **no es necesario** nunca. El hecho de la *anidación* no limita un ápice las posibilidades de la sintaxis.

Ver código fuente

PHP dispone de la función

show_source(*'pag'***);**

que permite visualizar el código fuente del *documento* que se indica en el parámetro *pag*.
Es muy útil para los propósitos de este curso, pero presenta un problema de **seguridad**.

Si escribes –en el parámetro *pag*– la dirección completa de una web cualquiera (que tenga extensión **php**) se visualizará su contenido, salvo que el PHP instalado en el servidor que la aloja la tenga expresamente desactivada.

Recuerda...

En HTML se puede asignar el color fondo a una celda incluyendo *bgcolor=RGB(x,y,z)* dentro de la etiqueta **<TD>**.

x, **y** ,**z** son los valores de las componentes de cada color primario.

```
                $columnas--;
        endwhile;
/* ¡muy importante!. Tendremos que reasignar a la variable columnas
      su valor inicial para que pueda ser utilizado en la proxima fila
          ya que el bucle (magenta) va reduciendo ese valor a cero
          y en caso de no restaurar el viejo valor no volvería a ejecutarse
          ya que no cumple la condición de ser mayor que cero */
    $columnas=3;
    echo "</TR>";
endwhile;
# por ultimo la etiqueta de cierre de la tabla
print "</table>";
?>
```

ejemplo40.php

Insertando condicionales en un bucle while

En este nuevo ejemplo hemos modificado ligeramente el anterior, incluyendo un condicional if. No incluimos aquí el código fuente para evitarte la monotonía de repetir íntegramente y con ligeras modificaciones el supuesto anterior.

Para visualizar ese código bastará que pulses en el enlace *Ver código fuente*. Las modificaciones que hemos introducido aparecen marcadas y podrás localizarlas rápidamente.

Utilizaremos a menudo esta forma de visualización del código fuente de los scripts. Es una opción de uso muy simple, utilizando la función **show_source()**, la que aprovechamos para comentar al margen.

Ver nuevo ejemplo | Ver código fuente

Ejercicio nº 22

Escribe un script –**ejercicio22.php**– en el que, mediante un bucle **while**, construya una tabla cuyas celdas tengan como *colores de fondo* una escala de grises que comience en RGB(0,0,0) y acabe en RGB(255,255,255) a intervalos de 5 unidades.

Recuerda que los diferentes tonos de grises se forman combinado valores iguales de los tres colores primarios.