La instrucción while

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

Ejemplo 1

```
        <?
                $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 importantísima
                # si no modificamos el valor de $A el bucle sería infinito
                        $A++;
                     print("</td></tr>");
                }
                # cerremos la etiqueta table
                print "</table>";
        ?>
```

Ejemplo 2

```
<?
        $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 $filas--
        # para escribir las etiquetas <tr> y </tr>  el modificador de la variable filas
        # y un segundo while (magenta) para insertar las etiquetas correspondientes a las celdas de cada fila
        while ($filas>0):
           echo "<tr>";
           $filas--;
              while ($columnas>0):
                    echo "<td>";
                    print "fila: ".$filas." columna: ".$columnas;
                    print ("</td>");
                    $columnas--;
              endwhile;
        /* ¡muy importante!. Tendremos que reasignar a la   su valor inicial para que pueda ser utilizado en la próxima 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>";  ?>
```

Ejercicio

Escribe un script **ejercicio.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.

```
<?
#utilicemoswhilesanidadosparaconstruirunatablade
$filas=5;$columnas=3;
#insertemoslaetiquetadeaperturadelatabla
print("<tableborder=2width=400align=center>");
#unprimerdowhilecomocondiciónquefilasseamayorquecero
#enestecaso,lavariabletendráquedisminuyendosuvalorcon$filas--
#paraescribirlasetiquetasy
#yelmodificadordelavariablefilas
#yunsegundosegundowhileparainsertarlasetiquetascorrespondientes
#alasceldasdecadafila

do{
echo"<tr>";
$filas--;
do{
echo"<td>";
print"fila:".$filas."columna:".$columnas;
print("</td>");
$columnas--;
}while($columnas>0);
/*muyimportantetendremosquereasignaralavariablecolumnas
suvalorinicialparaquepuedaserutilizadoenlaproximafila
yaqueelbucle(magenta)vareduciendoesevaloracero
yencasodenorestaurarelviejovalornovolveríaaejecutarse
yaquenocumplelacondicióndesermayorquecero*/
$columnas=3;
echo"</TR>";
}while($filas>0);

#porultimolaetiquetadecierredelatabla
print"</table>";
?>
```

El bucle do... while

Estamos ante una variante del bucle while que hemos visto en la página anterior. La sintaxis es la siguiente:

```
do {
        ...intrucción 1...
        .... instruccion2...
} while(condición) ;
```

Se diferencia de while en que en este caso se comprueba la condición después de haber ejecutado las instrucciones contenidas en el bucle, con lo cual, en el caso de que desde el comienzo no se cumplieran las condiciones establecidas en while, las instrucciones del bucle se ejecutarían una vez.

Respecto a la sintaxis, como puedes observar, detrás de do se inserta una llave ({) que señala el comienzo de las instrucciones pertenecientes al bucle. El final de esas instrucciones lo señala la otra llave (}) que precede a while(condición).

```
<?
        $A=0;
        do {
         ++$A;
         echo "Valores de A usando el do: ",$A,"<br>";
        } while($A<5);
        $B=7;
        do {
        echo "Pese a que B es mayor que 5 se ejecuta una vez.
        B= ",$B,"<br>";
        } while($B<5);
?>
```

Ejemplo 2

```
<?
        $A=500;
        do {
              if ($A>=500) {
         echo "No puede ejecutarse el bucle, porque no se cumple la condicion";
         break;
          }
         ++$A;
         echo "Valores de A usando el do: ",$A,"<br>";

        } while($A<500);
        echo "<BR>He salido del bucle porque A es: ",$A;
?>
```

La instrucción continue

Esta instrucción es aplicable tanto a bucles for como a los de tipo while o do while.

for

```
<?
        for ($i=0;$i<=10;$i++){

            #condición de  múltiplo de 2
          if ($i % 2 ==0 ) {
            continue ;
           }

          echo "La variable I vale ",$i,"<br>";
        }
 ?>
```

while

```
<?
        $i = 0;
        while ($i++ < 14) {

            #condición de no múltiplo de 3 usando para distinto la sintaxis !=

           if ($i % 3 !=0){
                continue ;
           }

           echo "El valor de i es: ",$i,"<br>";
        }

?>
```

do ... while

```
<?
        $i = 0;
        do {

            # condición de no múltiplo de 11. fíjate en la sintaxis alternativa
             # observa que aquí distinto lo hemos escrito <>

           if ($i % 11 <>0 ){
                continue ;
           }

           echo "El valor de i es: ",$i,"<br>";

        }while ($i++ < 100)
?>
```

Anidamientos 1

```
<?
$j=0;
while (++$j <5) {
    for($i=1;$i<5;$i++){

        if ($i==3){
          continue 2;
          }
       echo "El valor de j es: ",$j, " y el de i es: ",$i,"<br>";

    }
}
?>
```

Anidamientos 2

```
<?
        $j=0;$k=0;
        do {
        while (++$j <=5) {
            for($i=1;$i<=5;$i++){

            if ($i==2){
                continue 3;
                  }
            echo "El valor de k es: ",$k,
             " y el valor de j es: ",$j, " y el de i es: ",$i,"<br>";
                 }
            }
        }while ($k++ <=5);

?>
```

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 instrucciones contenidas entre { } serán ejecutadas cada vez que se reitere el bucle.

ejemplos de las diferentes variantes del bucle for:

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

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

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

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

?>
```

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

?>
```

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

        endfor;

?>
```

Sentencia goto , que te permite saltar directamente a otro punto del programa que indiques mediante una etiqueta.

```
<?php
        $a = 1;
        goto salto;
        $a++; //esta sentencia no se ejecuta
        salto:
        echo $a; // el valor obtenido es 1
?>
```

Ejemplos

```
<?php
        if ($a < $b) {
        print "a es menor que b";
        elseif ($a > $b)
        print "a es mayor que b";
        }
        else
        print "a es igual a b";
?>
```

```
<?php
        switch ($a) {
        case 0:
        print "a vale 0";
        break;
        case 1:
        print "a vale 1";
        break;
        default:
        print "a no vale 0 ni 1";
        }
?>
```

El bucle foreach

El bucle **foreach** es específico de los **array** y aplicable a ellos tanto si son *escalares* como si son de tipo *asociativo*.

Tiene dos posibles opciones. En una de ellas lee únicamente los valores contenidos en cada elemento del array. En el otro caso lee además los índices del array.

Utiliza la sintaxis:

```
foreach( array as var ){
        ...instrucciones...
}
```

donde *array* es el **nombre del array** (sin incluir índices ni corchetes), **as** es una *palabra* obligatoria y *var* el nombre de una variable (puede ser creada al escribir la instrucción ya que no requiere estar previamente definida).
Las instrucciones escritas *entre* las **{ }** permiten el tratamiento o visualización de los valores obtenidos.
La variable *var* no podrá ser utilizada para *guardar* valores. Hemos de tener en cuenta que su valor se *rescribe* en cada iteración del bucle y que al acabar este sólo contendrá el último de los valores leídos.

Lectura de índices y valores

Con una sintaxis como la que sigue se pueden leer no sólo los valores de un array sino también sus índices.

```
foreach( array as v1 => v2 ) {
        ...instrucciones...
}
```

donde *array* es el nombre de la matriz, **as** es una *palabra* obligatoria, *v1* es el nombre de la variable que recogerán los índices, los caracteres => (son obligatorios) son el separador entre ambas variables y, por último, *v2* es el nombre de la variable que recoge el valor de cada uno de los elementos del array.

Tanto esta función como la anterior realizan una *lectura secuencial* que comienza en el **primer valor** del array.

Ejemplo 1

```
<?
/* definimos un array escalar utilizando la sintaxis
  nombre del array=array (valores de los elemento separados por comas)
 si los valores son números no es necesario encerrarlos entre comillas */
$a=array("a","b","c","d","e");
/* definamos ahora un nuevo array, esta vez asociativo
   utilizando la sintaxis clave => valor tal como puedes ver */
$b=array(
 "uno"  =>"Primer valor",
 "dos"  =>"Segundo valor",
 "tres" =>"Tecer valor",
);
# establecemos el bucle que leerá el array $a
# recogiendo en la variable $pepe los valores extraídos
# y escribimos los valores
foreach($a as $pepe) {
echo $pepe,"<br>";
};
# repetimos el mismo proceso, ahora con $b que es un array asociativo
foreach($b as $pepe) {
echo $pepe,"<br>";
};
?>
```

Ejemplo 2

```
<?
      $a=array("a","b","c","d","e");
       $b=array(
       "uno"  =>"Primer valor",
       "dos"  =>"Segundo valor",
       "tres" =>"Tecer valor",
      );

      # en este caso extraeremos índices y valores de ambos arrays
      # usaremos $pepe para recoger los índices y $pepe para recoger los valores
      # y separaremos ambas variables por => que es el separador obligatorio
      # para estos casos

      foreach($a as $pepe=>$pepa) {
             echo "Indice: ",$pepe,"  Valor: ",$pepa,"<br>";
      };
      foreach($b as $pepe=>$pepa) {
             echo "Indice: ",$pepe,"  Valor: ",$pepa,"<br>";
      };
?>
```