

	<p style="text-align: center;">UNIVERSIDAD DON BOSCO FACULTAD DE INGENIERÍA ESCUELA DE COMPUTACIÓN</p>
<p style="text-align: center;">CICLO 01-2023</p>	<p style="text-align: center;">GUIA DE LABORATORIO N° 3</p>
	<p>Nombre de la práctica: Estructuras de control - Sentencias repetitivas Lugar de ejecución: Laboratorio de Informática Tiempo estimado: 2 horas Materia: Lenguajes Interpretados en el Servidor</p>

I. OBJETIVOS

En esta guía de práctica se espera que los estudiantes logren:

- Adquirir un amplio dominio de la sintaxis de cada una de las sentencias repetitivas disponibles en PHP.
- Hacer uso de sentencias para modificar el número de veces y la forma en qué se ejecuta un ciclo o lazo.

II. INTRODUCCION TEORICA

SENTENCIAS REPETITIVAS

Las sentencias repetitivas son un mecanismo proporcionado por el lenguaje PHP para poder repetir varias veces un bloque de instrucciones. La utilidad que se le da a este tipo de sentencias en programas específicos es la realización de conteos, realizar acumulación de valores en una variable, concatenar cadenas, construir los elementos de listas o tablas HTML, recorrer los elementos de un arreglo o de un objeto, etc.

PHP soporta cuatro tipos de sentencias repetitivas que son las mismas utilizadas por varios lenguajes de programación. Estos son: ***while***, ***do-while***, ***for*** y ***foreach***.

WHILE

Los bucles, lazos o ciclos *while* son los tipos de bucles más simples en PHP. Se comportan exactamente como en la mayoría de lenguajes de programación, tales como C/C++, Java, etc. La forma básica de una sentencia *while* es la siguiente:

```
while (expresion){[  
    [sentencia(s);]  
]}
```

Donde:

- **expresion**, es una expresión condicional que debe devolver un resultado lógico: *true* o *false* (verdadero o falso), y
- **sentencias**, es una instrucción o un bloque de instrucciones que se repetirán en tanto la expresión condicional se siga evaluando como verdadera.

NOTA: Cuando el ciclo o lazo *while* incluya una sola sentencia se pueden omitir las llaves. Sin embargo, es recomendable utilizarlas siempre.

El funcionamiento de una sentencia *while* es simple. Le dice a PHP que ejecute la(s) sentencia(s) que contiene repetidamente, mientras la expresión condicional del *while* se evalúe como verdadera (**true**). El valor de la expresión es comprobado cada vez al principio del bucle, así que incluso si este valor cambia durante la ejecución de la(s) sentencia(s) anidada(s), la ejecución no parará hasta el fin de la iteración (cada vez que PHP ejecuta las sentencias en el ciclo o bucle se da una iteración). Si la expresión *while* se evalúa como **false** desde el principio la(s) sentencia(s) anidada(s) no se ejecutarán ni siquiera una vez.

Al igual que con la sentencia *if*, existe una sintaxis alternativa para el ciclo o lazo *while*:

```
while (expresion):  
    [sentencia(s);]  
endwhile;
```

Los siguientes ejemplos son idénticos, y ambos imprimen números del 1 al 10:

```
/* ejemplo 1 */
```

```
$i = 1;  
while ($i <= 10) {  
    print $i++;  
    /* el valor impreso sería  
       $i antes del incremento  
       (post-incremento) */  
}
```

```
/* ejemplo 2 */
```

```
$i = 1;  
while ($i <= 10):  
    print $i;  
    $i++;  
endwhile;
```

DO ... WHILE

Los ciclos o bucles **do ... while** son muy similares a los bucles *while*, excepto que la condición se comprueba al final de cada iteración, en vez de al principio. La principal diferencia frente a los bucles regulares *while* es que en este, se garantiza la ejecución de la primera iteración de un bucle **do ... while** (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un bucle *while* regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como **false** desde el principio la ejecución del bucle finalizará inmediatamente).

Hay una sola sintaxis para los bucles **do...while**:

```
do{  
    sentencias;
```

} while (expresion);

Donde:

sentencias es el bloque de código a ejecutar un número indefinido de veces, dependiendo del resultado de la evaluación de la condición, y

expresion, es la expresión condicional que debe evaluarse para determinar si se seguirá ejecutando el bloque de sentencias del lazo.

El siguiente es un ejemplo:

```
$i = 0;
do {
    print $i;
} while ($i>0);
```

NOTA: Cuando el bucle *do-while* incluya una sola sentencia se pueden omitir las llaves. A pesar de ello es recomendable utilizarlas siempre.

FOR

Los ciclos o bucles **for** son los bucles más complejos en PHP por la construcción; sin embargo, son también los más utilizados porque su aplicación es más evidente. Se comportan exactamente igual que en lenguajes como C o Java. La sintaxis de un lazo **for** es:

```
for ([inicialización]; [condición]; [incremento]){  
    sentencia;  
}
```

Donde:

inicialización, es una expresión que inicializa una variable que funcionará como contador. Este valor representa el valor inicial de dicho contador a partir del cual comenzarán las iteraciones del lazo o bucle.

condicion, es una expresión condicional que se evaluará al inicio antes de ejecutar el bloque de sentencias. Si la evaluación de esta condición da por resultado *true*, el bloque de instrucciones se ejecuta, y si el resultado es *false* el bloque de instrucciones no se ejecuta.

incremento, es una sentencia que permite incrementar o decrementar el valor de la variable contador definida en la inicialización. El incremento|decremento puede ser de una unidad (caso más frecuente) o de varias unidades.

NOTA: Cuando el **for** incluya una sola sentencia se pueden omitir las llaves. Sin embargo, es recomendable utilizarlas siempre.

La primera expresión (**inicialización**) se evalúa (ejecuta) incondicionalmente una vez al principio del bucle y solamente esa vez.

Al comienzo de cada iteración, se evalúa la **condicion**. Si el resultado es *true*, el ciclo o lazo continúa y las sentencias anidadas se ejecutan. Si se evalúa como *false*, la ejecución del ciclo o lazo finaliza.

Justo, al final de cada iteración, se ejecuta la instrucción de **incremento|decremento**.

Cada una de las expresiones puede estar vacía. Que *expresion2* esté vacía significa que el bucle debería correr indefinidamente (PHP implícitamente lo considera como true, al igual que C). Esto puede que no sea tan inútil como se podría pensar, puesto que a menudo se quiere salir de un bucle usando una sentencia *break* condicional en vez de usar la condición de *for*.

Existe una sintaxis alternativa para la sentencia repetitiva *for*. Esta se muestra a continuación:

```
for([inicializacion]; [condicion]; [incremento]):  
    sentencia(s);  
endfor;
```

Considera los siguientes ejemplos. Todos ellos muestran números del 1 al 10:

```
<?php  
/* ejemplo 1 */  
  
for ($i = 1; $i <= 10; $i++) {  
    print $i;  
}  
  
/* ejemplo 2 */  
  
for ($i = 1; ;$i++) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
}  
  
/* ejemplo 3 */  
  
$i = 1;  
for (;;) {  
    if ($i > 10) {  
        break;  
    }  
    print $i;  
    $i++;  
}  
  
/* ejemplo 4 */  
  
for ($i = 1; $i <= 10; print $i, $i++) ;  
  
/* Ejemplo 5 */  
  
for($i = 1; $i <= 10; $i++):  
    print $i;  
endfor;  
  
?>
```

Por supuesto, el primer ejemplo parece ser el más elegante (o quizás el cuarto), pero uno puede descubrir que ser capaz de usar expresiones vacías en ciclos o lazos *for* resulta útil en algunas ocasiones.

INSTRUCCIONES BREAK Y CONTINUE.

Estas sentencias se utilizan dentro de los lazos, ciclos o bucles para provocar la terminación completa del lazo o el avance a la siguiente iteración del mismo.

Al utilizar la sentencia *break*, debe tener en cuenta que para que se ejecute debe producirse una situación en especial que amerite la terminación del ciclo o lazo. Para ello se puede utilizar una sentencia condicional, como se puede observar en el siguiente ejemplo:

```
/* ejemplo sentencia break */

$i = 1;
while ($i <= 10) {
    if($i%2 == 0){
        print "$i es par";
    }
    // Si el valor de $i es exactamente 5 el ciclo o lazo se terminará
    elseif($i == 5) {
        break;
    }
    else {
        print "$i es impar";
    }
}
```

El comportamiento de la sentencia *continue* es ligeramente distinto, ya que, en lugar de terminar por completo el ciclo o lazo, finaliza la ejecución de la iteración actual, pero continúa la ejecución del ciclo con la iteración siguiente, como se ilustra en el siguiente ejemplo:

```
/* ejemplo sentencia continue */

$i = 1;
while ($i <= 10) {
    if($i%2 == 0){
        print "$i es par";
    }
    //Si el valor de $i es exactamente 5 el ciclo o lazo avanzará a
    la siguiente iteración
    elseif($i == 5) {
        continue;
    }
    else {
        print "$i es impar";
    }
}
```

FOREACH

PHP4 y PHP5 incluyen la instrucción ***foreach***, tal como Perl y algunos otros lenguajes. Esta instrucción permite recorrer los elementos de una matriz de una forma sencilla. La instrucción permite obtener cada uno de los valores almacenados en el arreglo y asignarlos automáticamente en una variable para trabajar con ellos dentro del bloque de instrucciones del ***foreach***. También es posible obtener el índice si acaso, se ha utilizado un matriz asociativa. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

`foreach(expresion_array as $value) //sentencias;`

`foreach(expresion_array as $key => $value) //sentencias;`

La primera forma recorre el array dado por **`expresion_array`**. En cada iteración, el valor del elemento actual se asigna a **`$value`** y el puntero interno del array se avanza en una unidad (así en la siguiente iteración, se estará apuntando de forma automática el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable **`$key`** en cada iteración.

Notas:

- Cuando ***foreach*** comienza su primera ejecución, el puntero interno a la lista (*array*) se reinicia automáticamente al primer elemento del array. Esto significa que no se necesita llamar a **`reset()`** antes de un bucle ***foreach***.
- Hay que tener en cuenta que ***foreach*** trabaja con una copia de la lista (*array*) especificada y no la lista en sí, por ello el puntero de la lista no es modificado como en la construcción *each*.

Algunos ejemplos más para demostrar su uso:

```
<?php
/* foreach ejemplo 1: sólo valor*/
$a = array(1, 2, 3, 17);

foreach($a as $v) {
    print "Valor actual de \$a: $v.\n";
}

/* foreach ejemplo 2: valor (con clave impresa para ilustrar) */
$a = array(1, 2, 3, 17);

$i = 0; /* sólo para propósitos demostrativos */

foreach($a as $v) {
    print "\$a[$i] => $v.\n";
    $i++;
}

/* foreach ejemplo 3: clave y valor */
$a = array(
    "uno" => 1,
    "dos" => 2,
    "tres" => 3,
    "diecisiete" => 17
```

```

);
foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

/* foreach ejemplo 4: matriz multi-dimensional */

$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

/* foreach ejemplo 5: matriz dinámica */

foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
?>

```

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Material	Cantidad
1	Guía de práctica #3: Sentencias repetitivas	1
2	Computadora con WampServer instalado y funcionando correctamente	1
3	Editor PHP Sublime Text o Eclipse PHP	1
4	Memoria USB	1

IV. PROCEDIMIENTO

Indicaciones:

1. Asegúrese de digitar el código de los siguientes ejemplos que se presentan a continuación. Tenga en cuenta que los editores de código PHP no son compiladores solamente editores por lo tanto los errores de sintaxis lo podrá observar únicamente hasta que se ejecute el script al cargar la página en el navegador de su preferencia.
2. Descargar de la página de la universidad los recursos que utilizara en la guía
3. Recuerde crear una carpeta en la carpeta del servidor que está utilizando (www, httdocs), en donde almacenara los recursos de la guía y sus páginas PHP

Ejemplo 1: El siguiente ejercicio muestra cómo realizar la conversión de un número en sistema decimal a uno en sistema binario utilizando para ello cadenas y un ciclo do-while.

Archivo: convertir.php

```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <title>Conversión a binario</title>
5     <meta charset="utf-8" />
6     <link rel="stylesheet" href="css/decimalbinario.css" />
7     <script src="js/modernizr.custom.lis.js"></script>
8 </head>
9 <body>
10 <header>
11 <?php
12     if(isset($_POST['convertir']) && strlen($_POST['numero']) > 0){
13         $decimal = $_POST['numero'];
14     ?>
15     <h1><?=$decimal; ?><sub>10</sub> convertido a binario</h1>
16 <?php
17     }
18     else{
19         //No hacer nada
20     }
21 ?>
22 </header>
23 <div id="content">
24 <section>
25 <article>
26 <div id="wrapper">
27 <p>
28 <?php
29     if(isset($_POST['convertir'])) {
30         $decimal = $_POST['numero'];
31         if(strlen($decimal) > 0){
32             $msg = "El número decimal es: ";
33             $msg .= "<b>$decimal</b><br>\n";
34             echo $msg;
35             $binario = '';
36             do{
37                 $binario = $decimal % 2 . $binario;
38                 $msg = "$decimal % 2 = ";
39                 $msg .= "<b>$binario</b><br>\n";
40                 echo $msg;
41                 $decimal = (int)($decimal/2);
42             }while ($decimal > 0);
43             $msg = "<span class=\"marked\">Número binario resultante: ";
44             $msg .= "$binario</span>\n";
```



```

45         echo $msg;
46     }
47     else{
48         $prevpage = isset($_SERVER['HTTP_REFERER']) ? $_SERVER['HTTP_REFERER'] : "";
49         if(strlen($prevpage) == 0){
50             $prevpage = "decimalbinario.html";
51         }
52         $msg = "No ha ingresado dato en el campo de texto.<br />";
53         $msg .= "<a href=\"\" . $prevpage . \"\">Volver a intentar</a>";
54         echo $msg;
55     }
56 }
57 ?>
58 </p>
59 </div>
60 </article>
61 </section>
62 </div>
63 </body>
64 </html>

```

Resultado en el navegador:

125₁₀ convertido a binario

El número decimal es: 125

125	% 2	= 1
62	% 2	= 01
31	% 2	= 101
15	% 2	= 1101
7	% 2	= 11101
3	% 2	= 111101
1	% 2	= 1111101

Número binario resultante: 1111101

Ejemplo 2: El siguiente ejemplo muestra cómo crear una tabla de conversiones de monedas haciendo uso de un ciclo while.

Archivo: convmonedas.php

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <title>Convertidor de monedas</title>
5     <meta charset="utf-8" />
6     <link rel="stylesheet" href="css/monedas.css" />
7 </head>
8 <body>
9 <header>
10     <h2>Tabla de conversión de colones a dólares</h2><hr />
11 </header>
12 <section>
13 <article>
14 <?php
15     define("EQUIV","8.75");
16     $colones = 1.00;
17     $tabla = "<table>\n<thead>\n";
18     $tabla .= "<th>Colones</th>";
19     $tabla .= "<th>Dólares</th>";
20     $tabla .= "</thead>\n<tbody>\n";
21     while($colones <= 10){
22         $tabla .= "<tr>\n<td>&cent; ";
23         $tabla .= number_format($colones, 2, '.', ',') . "</td><td>\$ ";
24         $tabla .= number_format($colones / EQUIV, 2, '.', ',');
25         $colones += 0.25;
26         $tabla .= "</td>\n</tr>\n";
27     } // fin del while
28     $tabla .= "</tbody>\n</table>\n";
29     echo $tabla;
30 ?>
31 </article>
32 </section>
33 <script src="js/modernizr.custom.lis.js"></script>
34 </body>
35 </html>

```

Resultado en el navegador:

Tabla de conversión de colones a dólares



Colones	Dólares
₡ 1.00	\$ 0.11
₡ 1.25	\$ 0.14
₡ 1.50	\$ 0.17
₡ 1.75	\$ 0.20
₡ 2.00	\$ 0.23
₡ 2.25	\$ 0.26
₡ 2.50	\$ 0.29
₡ 2.75	\$ 0.31
₡ 3.00	\$ 0.34
₡ 3.25	\$ 0.37
₡ 3.50	\$ 0.40
₡ 3.75	\$ 0.43
₡ 4.00	\$ 0.46
₡ 4.25	\$ 0.49
₡ 4.50	\$ 0.51
₡ 4.75	\$ 0.54
₡ 5.00	\$ 0.57
₡ 5.25	\$ 0.60

Ejemplo 3: El siguiente ejemplo ilustra cómo utilizar ciclos o lazos do-while para acumular valores y obtener datos como el valor menor y mayor de una serie de números, así como el total de números pares presentes en la misma.

Archivo: intervalos.php

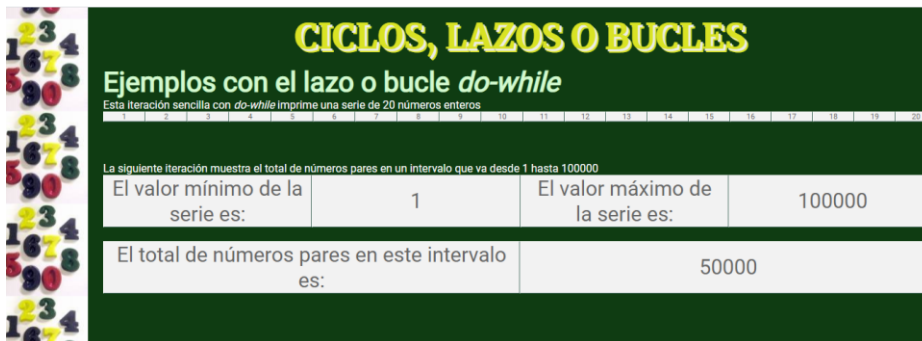
```
1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <title>Bucle do-while</title>
5     <meta charset="utf-8" />
6     <link rel="stylesheet" href="css/intervalos.css" />
7     <link rel="stylesheet" href="css/jquery-responsiveTables.css" />
8     <script src="js/modernizr.custom.lis.js"></script>
9     <script src="//ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js">
10 </script>
11 <script>window.jQuery || document.write('<script src="js/jquery-1.12.0.min.js">
12 </script>')
13 </script>
14 <script src="js/jquery-responsiveTables.js"></script>
15 <script src="js/jquery.responsiveText.js"></script>
16 <script>
17     $(document).ready(function() {
18         $('responsive').not('table').responsiveText();
19         $('table.responsive').responsiveTables();
20     });
21 </script>
22 </head>
23 <body>
24 <header>
25     <h1>Ciclos, lazos o bucles</h1>
26 </header>
27 <section>
28 <?php
29     $min = 1;
30     echo "<article>";
31     echo "<div id=\"main\">";
32     echo "<h2 class=\"subheading responsive\" data-compression=\"25\" data-min=\"14\"
33 data-max=\"40\">Ejemplos con el lazo o bucle <em>do-while</em></h2>";
34     echo "</article>";
35     echo "<p>\n";
36     echo "Esta iteración sencilla con ";
37     echo "<em>do-while</em> imprime una serie de ";
38     echo "20 números enteros\n";
39     echo "</p>\n";
40     echo "<table class=\"responsive\" data-min=\"11\" data-max=\"30\" cellpadding=\"0\"
41 cellspacing=\"0\">\n";
42     echo "<t<tbody>\n";
43     echo "<t<t<tr>\n";
44     do {
45         echo "<td>$min</td>\n";
46         $min++;
47     } while ($min <= 20);
48     echo "<t<t</tr>\n";
49     echo "</tbody>\n";
50     echo "</table>\n";
51     $min=1;
52     $max=100000;
53     echo "<p>&nbsp;</p>\n<p>&nbsp;</p>\n";
54     echo "<p>\nLa siguiente iteración muestra ";
55     echo "el total de números pares en un ";
56     echo "intervalo que va desde $min hasta $max\n<p>\n";
57     $contador=0;
```

```

58     if($max<$min){
59         $temp=$max;
60         $max=$min;
61         $min=$temp;
62     }
63     echo "<table class=\"responsive\" data-min=\"11\" data-max=\"30\" cellpadding=\"0\"
64     cellspacing=\"0\">\n";
65     echo "\t<tr>\n";
66     echo "\t\t<td>El valor mínimo de la serie es:</td>\n";
67     echo "\t\t<td>$min</td>\n";
68     echo "\t\t<td>El valor máximo de la serie es:</td>\n";
69     echo "\t\t<td>$max</td>\n";
70     echo "\t</tr>\n</table>\n";
71     do {
72         if($min%2 == 0) $contador++;$min++;
73     }while ($min <= $max);
74     echo "<table class=\"responsive\" data-min=\"11\" data-max=\"30\" cellpadding=\"0\"
75     cellspacing=\"0\">\n";
76     echo "\t<tr>\n<td>\n";
77     echo "El total de números ";
78     echo "pares en este intervalo es:\n</td>\n";
79     echo "\t\t<td>$contador</td>\n";
80     echo "\t</tr>\n";
81     echo "</table>\n";
82 ?>
83 </section>
84 </body>
85 </html>

```

Resultado en el navegador:



CICLOS, LAZOS O BUCLES

Ejemplos con el lazo o bucle *do-while*

Esta iteración sencilla con *do-while* imprime una serie de 20 números enteros

La siguiente iteración muestra el total de números pares en un intervalo que va desde 1 hasta 100000

El valor mínimo de la serie es:	1	El valor máximo de la serie es:	100000
El total de números pares en este intervalo es:	50000		

Ejemplo 4: En este ejercicio se ha utilizado la sintaxis alternativa de todas las estructuras de control (condicionales y repetitivas)

Archivo: factorial.php

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4     <title>Factorial de un número</title>
5     <meta charset="utf-8" />
6     <link rel="stylesheet" href="css/fonts.css" />
7     <link rel="stylesheet" href="css/factorial.css" />
8     <script src="js/modernizr.custom.lis.js"></script>
9     <script src="js/filterTextField.js"></script>
10 </head>
11 <body>
12 <header>
13     <h1>Obtener el factorial de un número</h1>
14 </header>
15 <section>
16 <article>
17 <div class="contenedor">
18 <?php
19     if(isset($_POST['enviar'])):
20         $cont = is_numeric($_POST['factorial']) ? $_POST['factorial'] : null;
21         $msg = isset($cont) || $cont==null ? "<p>No ha ingresado ningún número</p>" : "";
22         echo "<div class='encabezado'>";
23         if($msg == ""):
24             echo "No ha ingresado un número entero.";
25             echo "<a href='\"".$_SERVER['PHP_SELF']."' . \">Volver a intentarlo</a>";
26             exit(0);
27         endif;
28         if(!is_numeric($cont)):
29             echo "No ha ingresado un número entero.";
30             echo "<a href='\"".$_SERVER['PHP_SELF']."' . \">Volver a intentarlo</a>";
31             exit(0);
32         endif;
33         if($cont == 0):
34             $factorial = 1;
35             echo "0! = " . $factorial . "<br />";
36             exit(0);
37         endif;
38         $factorial = 1;
39         echo "<p>" . $cont . "! = ";
40         while($cont > 0):
41             $factorial *= $cont;
42             $cont--;
43         endwhile;
44         echo "<strong>" . $factorial . "</strong></p>";
45         echo "<p><a href='\"factorial.php\">Calcular el factorial de otro número</a>";
46         echo "</div>";
47     else:
48 ?>
49 <div class="encabezado">
50     Cálculo del factorial
51 </div>

```

```

52     <div class="formulario">
53     <form action="<?= $_SERVER['PHP_SELF'] ?>" method="POST">
54         <input type="text" name="factorial" id="factorial" placeholder="Número (entero)" />
55         <br>
56         <span id="numbersOnly">Sólo acepta números y punto decimal</span>
57         <div class="divisor"></div>
58         <input type="submit" value="Enviar" name="enviar" id="enviar" />
59     </form>
60 </div>
61 </div>
62
63 <?php
64     endif;
65 ?>
66 </article>
67 </section>
68 </body>
69 </html>

```

Resultado en el navegador:

OBTENER EL FACTORIAL DE UN NÚMERO

Cálculo Del Factorial

! 10

Enviar

OBTENER EL FACTORIAL DE UN NÚMERO

10! = 3628800

Calcular El Factorial De Otro Número

V. DISCUSION DE RESULTADOS

1. Realice un script PHP que mediante un formulario que solicite dos números, el primero de ellos entero o con parte decimal y el segundo necesariamente entero, calcule la potencia de elevar el primer número ingresado a la potencia dada por el segundo número. No puede utilizar la función pow() para resolver este problema, debe resolverlo haciendo uso de un ciclo o lazo en donde aproveche la característica de que un número elevado a una potencia es igual a multiplicar ese número por si mismo tantas veces como indique la potencia. Por ejemplo: $5^2 = 5 * 5 = 25$, o $3^4 = 3 * 3 * 3 * 3 = 81$.
2. Cree un script que le permita ingresar un número del 1 al 10 a través de un formulario, que solamente deberá contener un campo de texto, su etiqueta y un botón de envío. El script PHP que realizará deberá mostrar la tabla de multiplicar de ese número de forma ordenada y utilizando hojas de estilo para una buena apariencia visual.

VI. INVESTIGACION COMPLEMENTARIA

1. Investigue la utilización de las matrices unidimensionales y bidimensionales
2. Investigue que son las matrices asociativas
3. Investigue el uso de funciones para manejo de matrices con PHP

Para cada punto incluir un ejemplo en PHP

VII. BIBLIOGRAFIA

- Cabezas Granado, Luis Miguel. PHP 6 Manual Imprescindible. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España. 2010.
- Doyle, Matt. Fundamentos de PHP Práctico. Editorial Anaya Multimedia. 1ª. Edición. Madrid, España. 2010.
- Gutiérrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. Editorial Alfaomega RAMA. 1ra edición. México. Junio 2005.
- Gil Rubio, Francisco Javier/Villaverde, Santiago Alonso/Tejedor Cerbel, Jorge A. Creación de sitios web con PHP 5. Editorial McGraw-Hill. 1ra edición. Madrid, España, 2006.
- John Coggeshall. La Biblia de PHP 5. 1ra Edición. Editorial Anaya Multimedia. Madrid España.
- <http://www.php.net/manual/en>