

PROGRAMACIÓN II



SEMANA 03

ESTRUCTURAS DE CONTROL

Eric Gustavo Coronel Castillo
ecoronel@uch.edu.pe
gcoronelc.blogspot.com

ESTRUCTURAS DE CONTROL

Las estructuras de control permiten realizar la verificación de ciertas condiciones y en función a su resultado seleccionar las instrucciones a ejecutar o repetir su ejecución las veces que sea necesaria.

Índice

1	INTRODUCCIÓN.....	4
2	ESTRUCTURAS CONDICIONALES	5
2.1	CONDICIONAL SIMPLE: IF	5
2.2	CONDICIONAL DOBLE: IF – ELSE	8
2.3	CONDICIONAL MÚLTIPLE: IF – ELSEIF – ELSE	11
2.4	SELECTIVA MÚLTIPLE: SWITCH	15
3	ESTRUCTURAS REPETITIVAS	21
3.1	BUCLE: WHILE	21
3.2	BUCLE: DO – WHILE	26
3.3	BUCLE: FOR	28
3.4	BUCLE: FOREACH	31
3.5	INSTRUCCIONES: BREAK Y CONTINUE	35

1 Introducción

Las estructuras de control permiten bifurcar el flujo del programa y así ejecutar unas partes u otras del código según ciertas condiciones. PHP dispone de todas las estructuras clásicas de los lenguajes de alto nivel, con la sintaxis de C, C++ o Java, y además algunas otras estructuras más típicas de lenguajes interpretados como Perl o Bash.

En todos los casos, las estructuras de control contienen una expresión cuya evaluación como verdadero o falso determinará el flujo a seguir dentro de la estructura. Estas expresiones pueden ser una variable, una función (el valor que devuelve), una constante, o cualquier combinación de éstas con los operadores respectivos.

Estas estructuras están divididas en dos grupos:

Tipo	Descripción
Condicionales	En base a una condición se determina si se ejecuta o no un grupo de instrucciones. <ul style="list-style-type: none">- Condicional simple: <code>if</code>- Condicional doble: <code>if – else</code>- Condicional múltiple: <code>if – elseif – else</code>- Selectiva múltiple: <code>switch</code>
Repetitivas	Permiten repetir la ejecución de un grupo de instrucciones. <ul style="list-style-type: none">- Bucle: <code>while</code>- Bucle: <code>do – while</code>- Bucle: <code>for</code>- Bucle: <code>foreach</code>- Instrucciones: <code>break</code> y <code>continue</code>

2 Estructuras Condicionales

2.1 Condicional Simple: if

La instrucción `if` es la estructura de control más simple de todas. En su forma más simple sólo condiciona la ejecución de una o un grupo de sentencias.

Sintaxis:

```
if ( condición ) {  
  
    instrucciones;  
  
}
```

Si la **condición** se evalúa como verdadera, se ejecutan las instrucciones y después se sigue ejecutando el resto del programa. Si se evalúa como falsa, no se ejecutan las instrucciones y continúa con el resto del programa.

Ejemplo 1

Este programa muestra un formulario para que ingrese su nombre, este dato es enviado al programa `ejm02.php` para que sea procesado.

Archivo: `prog2\sem03\ejm01.html`

```
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
  </head>  
  <body bgcolor="#D2EBF7">  
    <form method = "post" action = "ejm02.php" accept-charset="utf-8" >  
      <table width="238">  
        <tr>  
          <td colspan="2">Tu Nombre</td>  
        </tr>  
        <tr>  
          <td width="143">  
            <input name="nombre" type="text" size="20" maxlength="20">  
          </td>  
          <td width="83">  
            <input type="submit" value="Enviar">  
          </td>  
        </tr>  
      </table>  
    </form>  
  </body>  
</html>
```

```
</tr>  
</table>  
</form>  
</body>  
</html>
```

El resultado:

Tu Nombre	
<input type="text" value="Gustavo Coronel"/>	<input type="button" value="Enviar"/>

Ejemplo 2

Este ejemplo recibe el nombre ingresado en el formulario del ejemplo **ejm01.html** y lo procesa.

Si no ingresa su nombre le muestra el mensaje **"Debes ingresar tu nombre."**, en caso contrario le muestra un saludo.

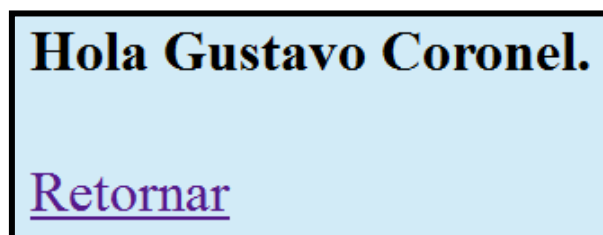
Archivo: prog2\sem03\ejm02.php

```
<?php
$nombre = null;
if( isset($_POST["nombre"]) ) {
    $nombre = trim( $_POST["nombre"] );
}

if( $nombre ) {
    $msg = "Hola $nombre.";
}
if( ! $nombre ) {
    $msg = "Debes ingresar tu nombre.";
}
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h4><?php echo($msg) ?></h4>
    <a href="ejm01.html">Retornar</a>
</body>
</html>
```

El resultado es:



2.2 Condicional Doble: if – else

A una instrucción **if** le podemos añadir instrucciones que se ejecuten cuando la condición es falsa mediante la cláusula **else**.

Sintaxis:

```
if ( condición ) {  
  
    instrucciones1;  
  
} else {  
  
    instrucciones2;  
  
}
```

Si **condición** se evalúa como verdadera, se ejecutan **instrucciones1**; si se evalúa como falsa, se ejecuta **instrucciones2**. En ambos casos luego se ejecuta el resto de instrucciones que sigan a la instrucción **if**.

Ejemplo 3

El siguiente ejemplo trata de un programa recursivo, inicialmente muestra un formulario para ingresar un número entero, luego evalúa si el número ingresado es par o impar, pero previamente verifica si realmente se ingresó un número entero.

Archivo: prog2\sem03\ejm03.php

```
<?php  
$mostrar = "formulario";  
if( isset( $_POST["seguro"] ) ) {  
    $mostrar = "resultado"; // Existe el dato  
}  
if( $mostrar == "resultado" ) {  
    // Obtener Dato  
    $n = $_POST["num"];  
    // Proceso  
    if(!is_numeric($n)){  
        $msg = "El dato ingresado no es un número";  
    } else {  
        if( (float)$n != (int)$n ) {  
            $msg = "El número ingresado no es un entero";  
        }  
    }  
}
```

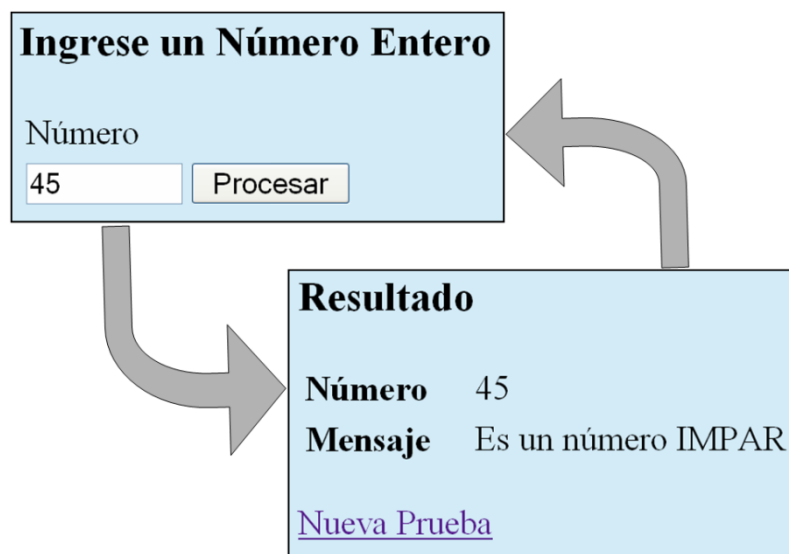


```
</tr>
</table>
<p><a href="ejm03.php">Nueva Prueba</a></p>
</div>
<?php } ?>
</body>
</html>
```

La etiqueta `div` se utiliza para demarcar las dos secciones, la del formulario y la del resultado, de esta manera es más fácil condicionar su impresión en el navegador de acuerdo el valor que tome la variable **\$mostrar**.

Note que todo el proceso se encuentra en la parte superior del programa y el código PHP puro, esto hace más fácil su lectura y por lo tanto su mantenimiento, como resultado se trata de minimizar el código espagueti a sólo instrucciones **echo(...)** dentro de etiquetas HTML.

Su funcionamiento se ilustra a continuación:



2.3 Condicional Múltiple: if – elseif – else

También podemos encadenar varias condiciones con la cláusula `elseif`.

Sintaxis:

```
if (condición1) {  
    instrucciones1;  
} elseif (condición2) {  
    instrucciones2;  
} elseif (condición3) {  
    instrucciones3;  
}  
...  
...  
elseif (condiciónN) {  
    instruccionesN;  
} else {  
    instruccionesElse;  
}
```

Ejemplo 4

El siguiente ejemplo muestra un formulario para el ingreso de dos notas, luego calcula el promedio y determina una condición según el siguiente cuadro:

Promedio	Condición
[18,20]	Excelente
[15,18>	Bueno
[11,15>	Regular
[6,11>	Malo
[0,6>	Pésimo

Archivo: prog2\sem03\ejm04.php

```
<?php  
$mostrar = "formulario";  
if( isset( $_POST["seguro"] ) ) {  
    $dato = "resultado"; // Existe el dato
```

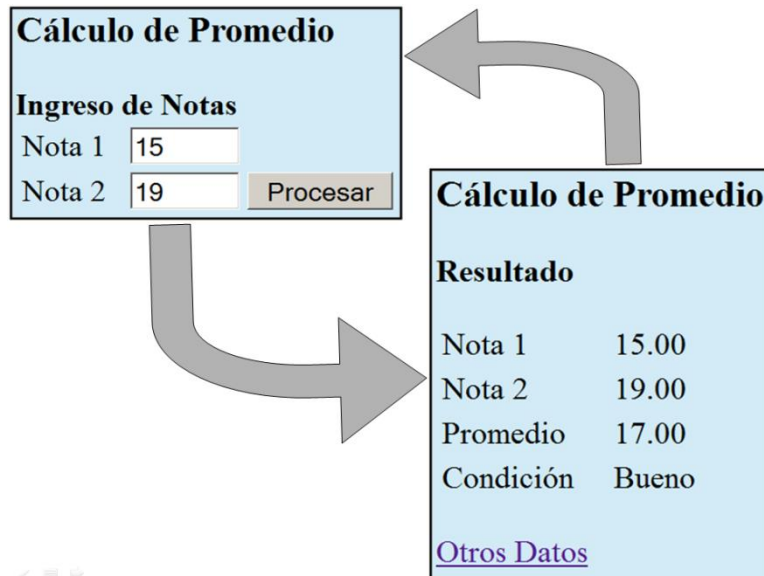


```
        </tr>
    </table>
</form>
</div>
<?php } ?>
<?php if( $mostrar == "resultado" ) { ?>
<div id="resultado">
    <p><strong>Resultado</strong></p>
    <table width="281">
        <tr>
            <td width="81">Nota 1</td>
            <td width="188"><?php echo($n1) ?></td>
        </tr>
        <tr>
            <td>Nota 2</td>
            <td><?php echo($n2) ?></td>
        </tr>
        <tr>
            <td>Promedio</td>
            <td><?php echo($pr) ?></td>
        </tr>
        <tr>
            <td>Condición</td>
            <td><?php echo($cond) ?></td>
        </tr>
    </table>
    <p><a href="ejm04.php">Otros Datos</a></p>
</div>
<?php } ?>
</body>
</html>
```

Al igual que en el Ejemplo 3 el código PHP que procesa el requerimiento se encuentra al inicio del programa, el código HTML también se encuentra bastante limpio, y el código espagueti sólo se limita a instrucciones **echo(...)** dentro del código PHP.

El valor de la variable **\$mostrar** determina si se muestra el formulario o se procesan los datos, su valor depende si existe o no el campo **seguro**.

Su funcionamiento se ilustra a continuación:



2.4 Selectiva Múltiple: switch

Esta estructura permite evaluar una expresión y según su resultado ejecutar un bloque de instrucciones.

Sintaxis:

```
switch (expresión) {  
    case valor1:  
        instrucciones1;  
        break;  
    case valor2:  
        instrucciones2;  
        break;  
    . . .  
    . . .  
    case valorN:  
        instruccionesN;  
        break;  
    default:  
        instruccionesDefault;  
}
```

Cuando se termina de ejecutar el código de un `case`, si no se finaliza el `switch` explícitamente con una instrucción `break`, se continúa ejecutando el código del siguiente `case` aunque no se cumpla la condición, hasta que se llegue al final del bloque `switch` o se finalice este con un `break`.

Ejemplo 5

Archivo: prog2\sem03\ejm05.php

```
<?php  
$num = rand(1,4);  
$msg = "";  
switch ($num) {  
    case 1:  
        $msg .= "Uno<br>";  
    case 2:  
        $msg .= "Dos<br>";  
    case 3:  
        $msg .= "Tres<br>";  
        break;
```

```

case 4:
    $msg .= "Cuatro<br>";
}
?>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body bgcolor="#D2EBF7">
    <table width="323">
      <tr>
        <td width="65"><strong>Número</strong></td>
        <td width="246"><?php echo($num) ?></td>
      </tr>
      <tr>
        <td valign="top"><strong>Mensaje</strong></td>
        <td><?php echo($msg) ?></td>
      </tr>
    </table>
  </body>
</html>

```

Un ejemplo de su ejecución se ilustra a continuación:

Número	2
Mensaje	Dos Tres

Si **\$num** toma el valor 1, se imprimirán las tres primeras cadenas; si toma el valor 2, la segunda y la tercera; si toma el valor 3, sólo la tercera y si toma el valor 4, sólo la última.

Ejemplo 6

El presente ejemplo genera un número entero entre 1 y 10, y luego verifica si es menor que 4, 4 o mayor que 4.

Archivo: prog2\sem03\ejm06.php

```
<?php
$num = rand(1,10);
switch ($num) {
    case 0: case 1: case 2: case 3:
        $msg = "Es menor que 4";
        break;
    case 4:
        $msg = "Es igual a 4";
        break;
    default:
        $msg = "Es mayor que 4";
        break;
}
?>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body bgcolor="#D2EBF7">
        <table width="323">
            <tr>
                <td width="65"><strong>Número</strong></td>
                <td width="246"><?php echo($num) ?></td>
            </tr>
            <tr>
                <td><strong>Mensaje</strong></td>
                <td><?php echo($msg) ?></td>
            </tr>
        </table>
    </body>
</html>
```

Un ejemplo de lo que podría ser el resulta es:

Número 7
Mensaje Es mayor que 4

Ejemplo 7

El siguiente ejemplo muestra un formulario para seleccionar un destino turístico, luego muestra un mensaje sobre el destino seleccionado.

Archivo: prog2\sem03\ejm07.php

```
<?php
$ciudad = null;
if( isset($_POST["ciudad"]) ) {
    $ciudad = $_POST["ciudad"];
    switch($ciudad){
        case "Chiclayo":
            $msg = "Ciudad de la amistad.";
            break;
        case "Trujillo":
            $msg = "Ciudad de la eterna primavera.";
            break;
        case "Cajamarca":
            $msg = "simplemente una ciudad espectacular.";
            break;
        case "Iquitos":
            $msg = "No te podrás olvidar de sus encantos";
            break;
        case "Huaraz":
            $msg = "Sus nevados son impresionantes.";
            break;
        case "Huancayo":
            $msg = "Ciudad INCONTRASTABLE.";
            break;
        case "Arequipa":
            $msg = "Lo mejor es su gente.";
            break;
        case "Cuzco":
            $msg = "Quedaras encantado.";
            break;
    }
}
?>

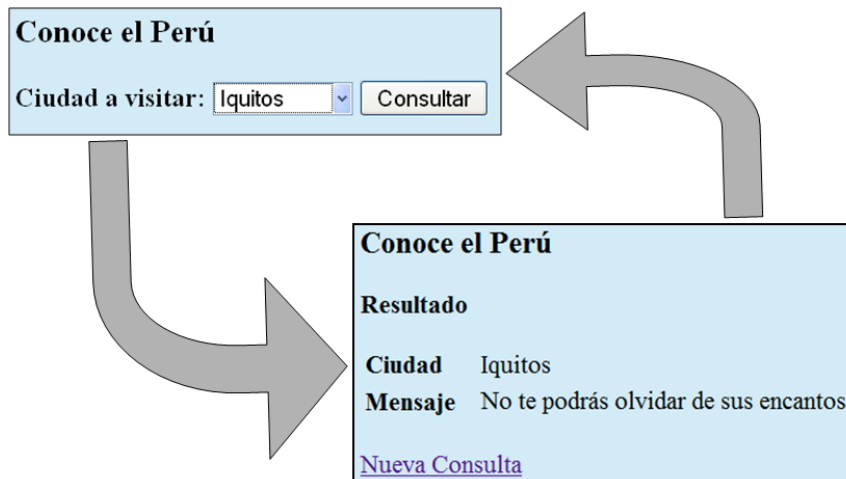
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h3>Conoce el Perú</h3>
```

```

<?php if($ciudad == null) { ?>
<div id="formulario">
    <form method = "post" action = "ejm07.php">
        <b>Ciudad a visitar:</b>
        <select size="1" name="ciudad">
            <option value="Chiclayo">Chiclayo</option>
            <option value="Trujillo">Trujillo</option>
            <option value="Cajamarca">Cajamarca</option>
            <option value="Iquitos">Iquitos</option>
            <option value="Huaraz">Huaraz</option>
            <option value="Huancayo">Huancayo</option>
            <option value="Arequipa">Arequipa</option>
            <option value="Cuzco">Cuzco</option>
        </select>
        <input type="submit" value="Consultar">
    </form>
</div>
<?php } ?>
<?php if($ciudad != null) { ?>
<div id="resultado">
    <p><strong>Resultado</strong></p>
    <table width="350">
        <tr>
            <td width="68"><strong>Ciudad</strong></td>
            <td width="270"><?php echo($ciudad) ?></td>
        </tr>
        <tr>
            <td><strong>Mensaje</strong></td>
            <td><?php echo($msg) ?></td>
        </tr>
    </table>
    <br/>
    <a href="ejm07.php">Nueva Consulta</a>
</div>
<?php } ?>
</body>
</html>

```

Su funcionamiento se ilustra a continuación:



La etiqueta `div` se está utilizando para agrupar las secciones que se deben mostrar como una sola unidad, en este caso tenemos dos, la sección del formulario y la sección del resultado.

La impresión de una u otra sección está condicionada al valor que toma la variable **\$ciudad**, si toma valor **null** se muestra la sección del formulario, en caso contrario se muestra la sección del resultado.

3 Estructuras Repetitivas

3.1 Bucle: while

Representa una estructura que ejecuta un grupo de instrucciones mientras una condición es verdadera. Para ingresar al bucle inicialmente la condición debe ser verdadera, de lo contrario nunca se ejecuta el bucle.

Sintaxis:

```
while ( condición ) {  
  
    instrucciones;  
  
}
```

Ejemplo 8

El siguiente programa muestra un formulario para ingresar dos números enteros, luego muestra los números comprendidos entre estos dos números y un mensaje que indica si es o no múltiplo de 3.

Archivo: prog2\sem03\ejm08.php

```
<?php  
$mostrar = "formulario";  
if( isset( $_POST["procesar"] ) ) {  
    $mostrar = "resultado";  
    $num1 = $_POST["num1"];  
    $num2 = $_POST["num2"];  
}  
?>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
    </head>  
    <body bgcolor="#D2EBF7">  
        <?php if( $mostrar == "formulario" ) { ?>  
            <div id="formulario">  
                <h3>Dos Números Enteros</h3>  
                <form method = "post" action = "ejm08.php">  
                    Número 1:  
                    <input name="num1" type="text" size="8" maxlength="3">
```

```

        <br>
        Número 2:
        <input name="num2" type="text" size="8" maxlength="3">
        <input type="submit" value="Procesar" name="procesar">
    </form>
</div>
<?php } ?>
<?php if( $mostrar == "resultado" ) { ?>
<div id="resultado">
<table border='1' width='216'>
    <tr>
        <th width="90">Número</th>
        <th width="110">Múltiplo de 3</th>
    </tr>
    <?php
    while ($num1 <= $num2){
        $msg = ($num1%3 == 0)?"Si":"No";
    ?>
        <tr>
            <td align="center"><?php echo($num1) ?></td>
            <td align="center"><?php echo($msg) ?></td>
        </tr>
        <?php
        $num1++;
    }
    ?>
</table>
<br>
<a href='ejm08.php'>Nueva Prueba</a>
</div>
<?php } ?>
</body>
</html>

```

Para este ejemplo la variable **\$mostrar** es la que determina que sección es la que se debe mostrar, su valor por defecto es **formulario**, cuando se determina que existe el campo **procesar** cambia a **resultado**.

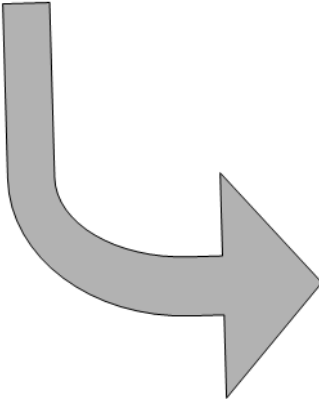
El bucle **while** se utiliza para repetir la impresión de la segunda fila de la tabla, y la variable **\$num1** está actuando como contador.

El resultado se ilustra a continuación:

Dos Números Enteros

Número 1:

Número 2:



Número	Múltiplo de 3
14	No
15	Si
16	No
17	No
18	Si
19	No

[Nueva Prueba](#)

Ejemplo 9

El siguiente ejemplo muestra un formulario para el ingreso del mes y día del año 2009, luego debe calcular los días transcurridos desde el 1ro de enero hasta la fecha ingresada del mismo año y determinar a qué día de la semana pertenece.

Archivo: prog2\sem03\ejm09.php

```
<?php
$mostrar = "formulario";
if( isset( $_POST["procesar"] ) ) {
    $mostrar = "resultado";
    $mes = $_POST["mes"];
    $dia = $_POST["dia"];
    $dt = $dia; // Acumulador de dias transcurridos
    $k = 1; // contador
    while ($k < $mes){
        switch($k){
            case 1: case 3: case 5: case 7:
            case 8: case 10: case 12:
                $dt += 31;
                break;
            case 4: case 6: case 9: case 11:
                $dt += 30;
                break;
            case 2:
                $dt += 28;
                break;
        }
        $k++;
    }
    $r = $dt % 7;
    switch($r){
        case 0: $ds = "Miercoles"; break;
        case 1: $ds = "Jueves"; break;
        case 2: $ds = "Viernes"; break;
        case 3: $ds = "Sábado"; break;
        case 4: $ds = "Domingo"; break;
        case 5: $ds = "Lunes"; break;
        case 6: $ds = "Martes"; break;
    }
}
?>
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

```

</head>
<body bgcolor="#D2EBF7">
  <?php if( $mostrar == "formulario" ){ ?>
    <h2>Ingrese un mes y día<br>
    del año 2009</h2>
    <form method = "post" action = "ejm09.php">
      Mes: <input name="mes" type="text" size="5" maxlength="2">
      Día: <input name="dia" type="text" size="5" maxlength="2">
      <input type="submit" value="Procesar" name="procesar">
    </form>
  <?php } ?>
  <?php if( $mostrar == "resultado" ) { ?>
    <div id="resultado">
      <h2>Resultado</h2>
      <table width="323">
        <tr>
          <td width="144"><strong>Días Transcurridos</strong></td>
          <td width="167"><?php echo($dt) ?></td>
        </tr>
        <tr>
          <td><strong>Día de la Semana</strong></td>
          <td><?php echo($ds) ?></td>
        </tr>
      </table>
      <p><a href='ejm09.php'>Nueva Prueba</a></p>
    </div>
  <?php } ?>
</body>
</html>
    
```

Su ejecución se ilustra a continuación:



3.2 Bucle: do – while

La estructura `do - while` también representa un bucle que se ejecuta mientras la condición es verdadera, pero la comprobación se realiza después de la primera iteración, como consecuencia de ello el bucle se ejecuta por lo menos una vez.

Sintaxis

```
do {  
  
    instrucciones;  
  
} while ( condición );
```

Ejemplo 10

El siguiente ejemplo permite obtener la tabla de multiplicar de un número ingresado a través de un formulario.

Archivo: prog2\sem03\ejm10.php

```
<?php  
$mostrar = "formulario";  
if( isset( $_POST["procesar"] ) ) {  
    $mostrar = "resultado";  
    $num = $_POST["num"];  
}  
?>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
    </head>  
    <body bgcolor="#D2EBF7">  
        <h2>Tabla de Multiplicar</h2>  
        <?php if( $mostrar == "formulario" ){ ?>  
        <form method = "post" action = "ejm10.php">  
            Número <input name="num" type="text" size="5" maxlength="2" id="num">  
            <input type="submit" value="Procesar" name="procesar">  
        </form>  
        <?php } ?>  
        <?php if( $mostrar == "resultado" ) { ?>  
        <div id="resultado">  
            <h3>Tabla del <?php echo($num) ?></h3>  
            <table border="1">
```

```

<?php
$k = 0;
do{
    $k++;
    $r = $k * $num;
    ?>
    <tr>
        <td width="30" align="center"><?php echo($num) ?></td>
        <td width="30" align="center">*</td>
        <td width="30" align="center"><?php echo($k) ?></td>
        <td width="30" align="center">=</td>
        <td width="30" align="center"><?php echo($r) ?></td>
    </tr>
    <?php } while( $k < 12 ); ?>
</table>
<p><a href='ejm10.php'>Nueva Prueba</a>    </p>
</div>
<?php } ?>

</body>
    
```

El resultado se ilustra a continuación:

Tabla de Multiplicar

Número

←

Tabla de Multiplicar

Tabla del 5

5	*	1	=	5
5	*	2	=	10
5	*	3	=	15
5	*	4	=	20
5	*	5	=	25
5	*	6	=	30
5	*	7	=	35
5	*	8	=	40
5	*	9	=	45
5	*	10	=	50
5	*	11	=	55
5	*	12	=	60

[Nueva Prueba](#)

→

3.3 Bucle: for

El bucle `for` no es estrictamente necesario, puede ser sustituido fácilmente por un bucle `while`. Sin embargo, el bucle `for` resulta muy útil cuando debemos ejecutar un bloque de código a condición de que una variable se encuentre entre un valor mínimo y otro máximo. Su estructura es la misma que en C ó Java.

Sintaxis

```
for (expresión1; expresión2; expresión3) {  
  
    instrucciones;  
  
}
```

Donde:

Elemento	Descripción
expresión1	Es la iniciación del bucle. Generalmente da un valor inicial a una o varias variables (separadas por comas). Sólo se ejecuta una vez, al principio, cuando el flujo del programa llega al bucle.
expresión2	Es la condición. Mientras que expresión2 sea verdadera, el bucle estará iterando. Se evalúa al inicio de cada iteración, y si no es verdadera la siguiente iteración ya no se realiza y finaliza el bucle, continuando la ejecución del programa con el resto del código de después del <code>for</code> .
expresión3	Es el paso de iteración. Se ejecuta después de cada iteración, y generalmente modifica el valor de alguna variable, separadas por comas si hay más de una.

Cualquiera de las tres expresiones puede estar vacía, aunque en este caso debemos tener cuidado de realizar su función en el cuerpo del bucle.

Diagrama de flujo de la estructura `for`

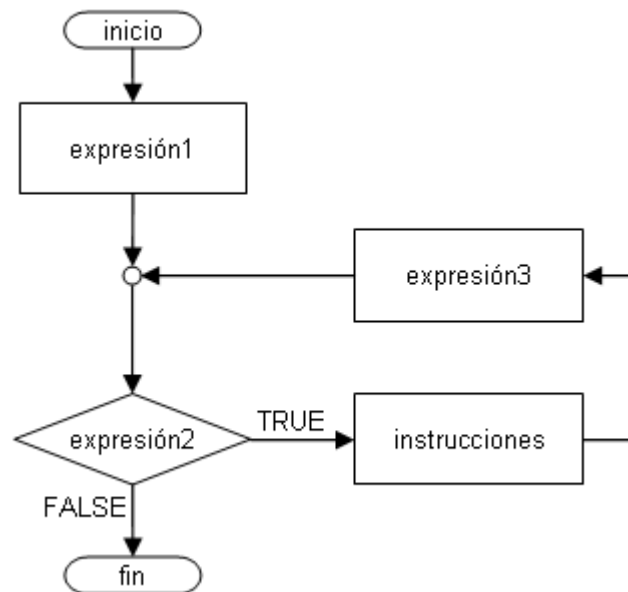


Figura 1 Diagrama de Flujo de la estructura `for`.

Ejemplo 11

El siguiente ejemplo permite calcular el factorial de un número, inicialmente presenta un formulario para el ingreso de un número, luego muestra el resultado.

Archivo: prog2\sem03\ejm11.php

```

<?php
$mostrar = "formulario";
if( isset( $_POST["procesar"] ) ) {
    $mostrar = "resultado";
    $num = $_POST["num"];
    $fact = 1;
    for ($i = 2; $i <= $num; $i++ ) {
        $fact *= $i;
    }
}
?>

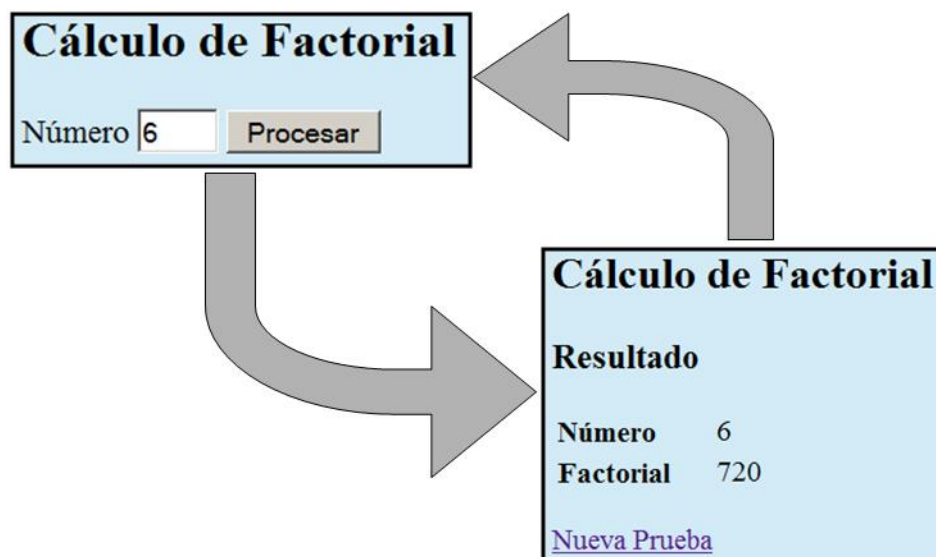
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h2>Cálculo de Factorial</h2>
    <?php if( $mostrar == "formulario" ){ ?>
    <form method = "post" action = "ejm11.php">

```

```

        Número <input type="text" name="num" size="2" maxlength="2">
        <input type="submit" value="Procesar" name="procesar">
    </form>
    <?php } else {?>
    <div id="resultado">
        <h3>Resultado</h3>
        <table width="189">
            <tr>
                <td width="82"><strong>Número</strong></td>
                <td width="95"><?php echo($num) ?></td>
            </tr>
            <tr>
                <td><strong>Factorial</strong></td>
                <td><?php echo($fact) ?></td>
            </tr>
        </table>
        <p><a href="ejm11.php">Nueva Prueba</a></p>
    </div>
    <?php } ?>
</body>
</html>
    
```

Un ejemplo del funcionamiento del programa se ilustra a continuación:



3.4 Bucle: foreach

El bucle `foreach` representa una estructura de control típica de lenguajes interpretados como `Perl` y `Bash`, permite hacer el recorrido a través de los elementos de una matriz.

Sintaxis 1

La siguiente sintaxis se aplica sobre arreglos simples.

```
foreach (array as $variable) {  
  
    instrucciones;  
  
}
```

Ejemplo 12

El siguiente ejemplo hace el recorrido a través de un arreglo y muestra cada uno de sus elementos.

Archivo: `prog2\sem03\ejm12.php`

```
<?php  
// Generación del Array  
$size = rand(5, 15);  
for($k = 1; $k <= $size; $k++){  
    $list[] = rand(20, 50);  
}  
?>  
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
  </head>  
  <body bgcolor="#D2EBF7">  
    <h3>Recorrido de un array</h3>  
    <table width="500">  
      <tr>  
        <td width="77"><strong>Tamaño</strong></td>  
        <td width="411"><?php echo($size) ?></td>  
      </tr>  
      <tr>  
        <td><strong>Elementos</strong></td>  
        <td>  
          <?php // Imprimir el arreglo
```

```
        foreach ($list as $value) {  
            echo("$value ");  
        }  
    ?>  
</td>  
</tr>  
</table>  
<p>&nbsp;</p>  
</body>  
</html>
```

El resultado:

Recorrido de un Array

Tamaño 6

Elementos 50 22 30 47 44 49

Sintaxis 2

La siguiente sintaxis se aplica sobre arreglos asociativos.

```
foreach( array as $clave => $valor) {  
  
    instrucciones;  
  
}
```

Ejemplo 13

En el siguiente ejemplo se ilustra el recorrido de un arreglo asociativo, mostrando tanto la clave y su valor.

Archivo: prog2\sem03\ejm13.php

```
<?php  
// Generación del Array  
$list = array(  
    "Chiclayo" => "Amistad",  
    "Trujillo" => "Primavera",  
    "Cuzco" => "Machu Picchu",  
    "Arequipa" => "Ciudad Blanca"  
);  
?>  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
    </head>  
    <body bgcolor="#D2EBF7">  
        <h3>Recorrido de un Array Asociativo</h3>  
        <table border="1">  
            <tr>  
                <th width="130" scope="col">Clave</th>  
                <th width="130" scope="col">Valor</th>  
            </tr>  
            <?php foreach ($list as $key => $value) { ?>  
                <tr>  
                    <td><?php echo($key) ?></td>  
                    <td><?php echo($value) ?></td>  
                </tr>  
            <?php } ?>  
        </table>
```



```
<p>&nbsp;</p>
</body>
</html>
```

Resultado:

Recorrido de un Array Asociativo	
Clave	Valor
Chiclayo	Amistad
Trujillo	Primavera
Cuzco	Machu Picchu
Arequipa	Ciudad Blanca

3.5 Instrucciones: break y continue

La sentencia `break` nos permite salir inmediatamente de una estructura de control `switch`, `while`, `do – while`, `for` o `foreach`.

Por su parte, la instrucción `continue` lo que hace es saltarse el resto de la iteración actual, y pasar directamente a la siguiente.

Ejemplo 14

En el siguiente ejemplo el bucle `while` finaliza en la iteración `$k == 5`, pese a que la condición del `while` es `$k <= 10`.

Archivo: prog2\sem03\ejm14.php

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
  <h4>Ejemplo de BREAK</h4>
  <?php
    $k = 1;
    while ($k <= 10) {
      echo "\$k = $k <br>";
      if( $k == 5) {
        break;
      }
      $k++;
    }
  ?>
</body>
</html>
```

Cuando la variable `$k` toma el valor `5` se ejecuta la instrucción `break`, y por lo tanto finaliza el bucle.

El resultado:

Ejemplo de BREAK

```
$k = 1  
$k = 2  
$k = 3  
$k = 4  
$k = 5
```

La instrucción `break` tiene un parámetro en el que especificamos el número de niveles de bucles anidados de los que queremos salir. Por defecto es uno (salir del bucle más interno), tal como se aprecia en el siguiente esquema:

```
$a = 0;  
while ($a < 10) {  
    $b = 0;  
    -----  
    -----  
    while ($b < 5) {  
        -----  
        -----  
        if ($b == 2) {  
            break; // Equivale a "break 1"  
        }  
    }  
  
    while ($b < 5) {  
        -----  
        -----  
        if ($a == 3 && $b == 3) {  
            break 2; // Saldría de los DOS bucles.  
        }  
    }  
}
```

Ejemplo 15

En el siguiente ejemplo se saltaría la instrucción `echo` de la iteración cuando `$k` toma el valor **5**.

Archivo: prog2\sem03\ejm15.php

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body bgcolor="#D2EBF7">
    <h4>Ejemplo de CONTINUE</h4>
    <?php
      $k = 0;
      while ($k < 8) {
        $k++;
        if( $k == 5) {
          continue;
        }
        echo "\$k = $k <br>";
      }
    ?>
  </body>
</html>
```

El resultado:

Ejemplo de CONTINUE

```
$k = 1
$k = 2
$k = 3
$k = 4
$k = 6
$k = 7
$k = 8
```

Ejemplo 16

El siguiente ejemplo evalúa si el número ingresado a través de un formulario es primo ó no.

Archivo: prog2\sem03\ejm16.php

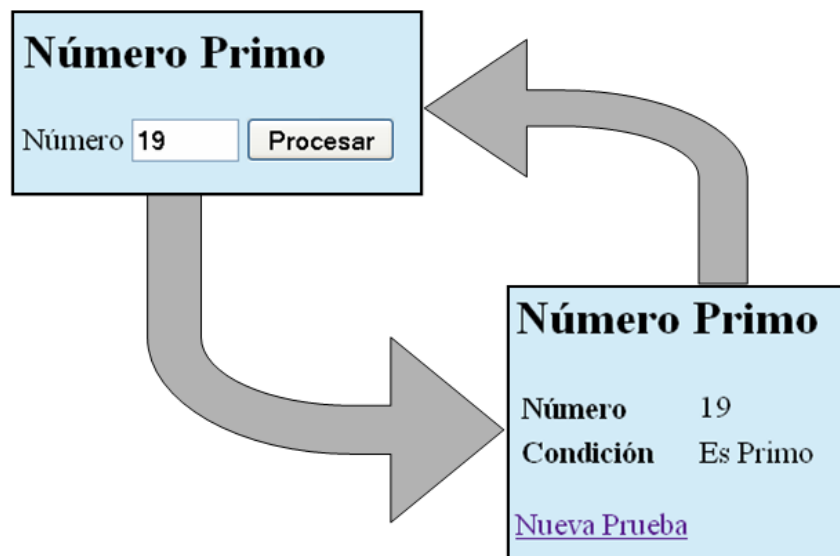
```
<?php
$mostrar = "formulario";
if( isset( $_POST["procesar"] ) ) {
    $mostrar = "resultado";
    $num = (int)$_POST["num"];
    $esPrimo = TRUE;
    $k = 2;
    while ($k < $num) {
        if (($num % $k) == 0) {
            $esPrimo = FALSE;
            break;
        }
        $k ++;
    }
    if ($esPrimo){
        $msg = "Es Primo";
    } else {
        $msg = "No es Primo";
    }
}
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h2>Número Primo</h2>
    <?php if( $mostrar == "formulario") { ?>
    <form method = "post" action = "ejm16.php">
        Número
        <input type="text" name="num" size="5" maxlength="2">
        <input type="submit" value="Procesar" name="procesar">
    </form>
    <?php } else { ?>
    <div>
        <table width="264">
            <tr>
                <td width="86"><strong>Número</strong></td>
                <td width="166"><?php echo($num) ?></td>
```

```

    </tr>
    <tr>
        <td><strong>Condición</strong></td>
        <td><?php echo($msg) ?></td>
    </tr>
</table>
<p><a href='ejm16.php'>Nueva Prueba</a> </p>
</div>
<?php } ?>
</body>
</html>
    
```

El resultado:



Ejemplo 17

El siguiente ejemplo muestra los N primeros términos de la serie de Fibonacci, el valor de N se ingresa a través de un formulario.

Archivo: prog2\sem03\ejm17.php

```
<?php

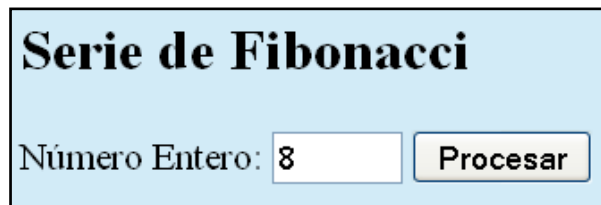
$mostrar = FALSE;
if( isset($_POST["procesar"]) ) {
    $mostrar = TRUE;
    $num = $_POST["num"];
    if ($num <= 2){
        $msg = "Debe ingresar un número entero mayor que 2.";
    } else {
        $msg = "1 1";
        $a = 1;
        $b = 1;
        for( $k = 3; $k <= $num; $k++ ) {
            $c = $a + $b;
            $msg .= " $c";
            $a = $b;
            $b = $c;
        }
    }
}

?>

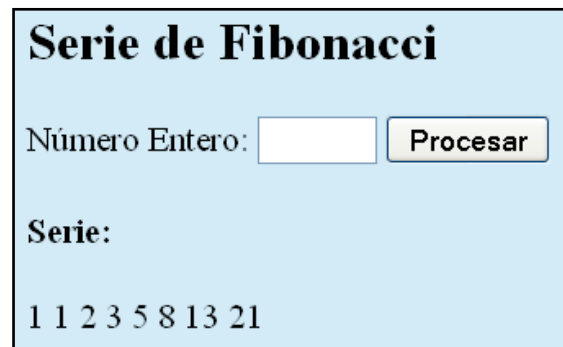
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h2>Serie de Fibonacci</h2>
    <form method = "post" action = "ejm17.php">
        Número Entero:
        <input type="text" name="num" size="5" maxlength="2">
        <input type="submit" value="Procesar" name="procesar">
    </form>
    <?php if( $mostrar ) { ?>
    <div id="resultado">
        <h4>Serie:</h4>
        <p><?php echo($msg) ?></p>
    </div>
```

```
<?php } ?>  
</body>  
</html>
```

Cuando ejecuta éste programa inicialmente se muestra la siguiente interfaz:



Cuando hacemos clic en el botón **Procesar** obtenemos el siguiente resultado:



Se puede ejecutar otra prueba ingresando otro número entero y haciendo clic nuevamente en el botón **Procesar**.

La variante con los ejemplos anteriores es que la sección del formulario siempre está visible.

Ejemplo 18

Desarrollar un programa que simule las operaciones de una cuenta de ahorro, el saldo inicial debe ser 5000.

Para mantener el estado del saldo utilizaremos un campo oculto de nombre **saldo**, que también servirá para saber si se trata de la primera ejecución o ya estamos realizando alguna transacción.

Archivo: prog2\sem03\ejm18.php

```
<?php
if( isset($_POST["saldo"]) ){
    $saldo = (double)$_POST["saldo"];
    $tipo = $_POST["tipo"];
    $importe = (integer)$_POST["importe"];
    $msg = "Saldo Actual: " . number_format($saldo,2,".",",") . "<br>";
    $msg .= "Tipo: $tipo <br>";
    $msg .= "Importe: " . number_format($importe,2,".",",") . "<br>";
    if( $tipo == "D" ) {
        $saldo += $importe;
        $msg .= "Nuevo Saldo: " . number_format($saldo,2,".",",") . "<br>";
    } else {
        if( ($saldo - $importe) < 0 ) {
            $msg = "Operación no es posible. <br>";
            $msg .= "No tiene fondos suficientes. <br>";
        } else {
            $saldo -= $importe;
            $msg .= "Nuevo Saldo: " . number_format($saldo,2,".",",") . "<br>";
        }
    }
} else {
    $saldo = 5000.0; // Saldo Inicial
}
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h2>Movimiento Bancario</h2>
    <h3>Saldo Actual: <?php echo $saldo; ?></h3>
    <form method="post" action="ejm18.php">
        <input type="hidden" name="saldo" value="<?php echo $saldo; ?>">
        <table width="248">
            <tr>
```

```

        <td width="60">Tipo: </td>
        <td width="176">
            <select name="tipo" size="1">
                <option value="D">D - Depósito</option>
                <option value="R">R - Retiro</option>
            </select>
        </td>
    </tr>
    <tr>
        <td>Importe: </td>
        <td>
            <input type="text" name="importe" size="8" maxlength="4">
        </td>
    </tr>
    <tr>
        <td colspan="2">
            <input type="submit" value="Enviar">
            <input type="reset" value="limpiar"> </td>
        </tr>
    </table>
</form>
<?php
if( isset($msg) ) {
    echo("<hr size='2'>");
    echo($msg);
    echo("<hr size='2'>");
}
?>
</body>
</html>
    
```

La ejecución muestra inicialmente la siguiente interfaz:



Luego de seleccionar un tipo de movimiento e ingresar el importe, hacemos clic sobre el botón **Procesar**, obtenemos el siguiente resultado:

Movimiento Bancario

Saldo Actual: 5100

Tipo:

Importe:

Saldo Actual: 5,000.00

Tipo: D

Importe: 100.00

Nuevo Saldo: 5,100.00

Ejemplo 19

Desarrollar un programa para encontrar los divisores de un número entero, o indicar si se trata de un número primo.

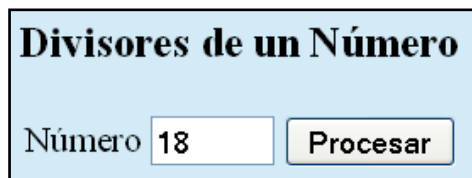
Archivo: prog2\sem03\ejm19.php

```
<?php
$resultado = FALSE;
if( isset($_POST["procesar"]) ) {
    $resultado = TRUE;
    $num = $_POST["num"];
    $msg = "";
    for( $j = 2; $j < $num; $j++ ) {
        if( ($num % $j) == 0 ) { $msg .= "$j "; }
    }
    if( $msg == "" ) {
        $msg = "Es un número primo.";
    }
}
?>

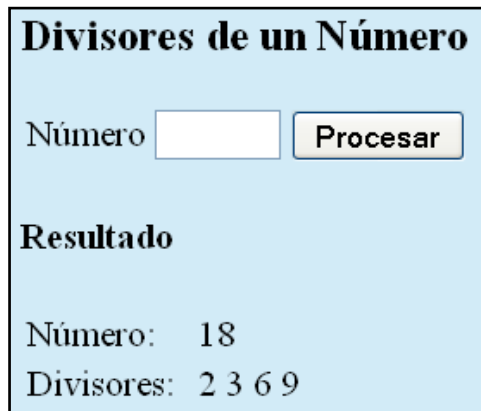
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h3>Divisores de un Número</h3>
    <form method = "post" action = "ejm19.php">
        <table width="185">
            <tr>
                <td width="60">Número</td>
                <td width="42">
                    <input type="text" name="num" size="5"
                        maxlength="3" id="num">
                </td>
                <td width="67">
                    <input name="procesar" type="submit"
                        id="procesar" value="Procesar">
                </td>
            </tr>
        </table>
    </form>
    <?php if( $resultado ) { ?>
    <div id="resultado">
        <h4>Resultado</h4>
```

```
<table width="454">
  <tr>
    <td width="70">Número:</td>
    <td width="372"><?php echo($num) ?></td>
  </tr>
  <tr>
    <td>Divisores:</td>
    <td><?php echo($msg) ?></td>
  </tr>
</table>
</div>
<?php } ?>
</body>
</html>
```

Cuando ejecutamos el programa inicialmente obtenemos la siguiente interfaz:



Luego de ingresar un número entero y hacer clic en el botón **Procesar** obtenemos el siguiente resultado:



Para analizar otro número debe ingresarlo en el campo respectivo y nuevamente hacer clic en el botón **Procesar**.

Ejemplo 20

Desarrollar un programa que permita determinar si una cadena es un palíndromo, un palíndromo es un texto que se lee igual de derecha a izquierda que de izquierda a derecha.

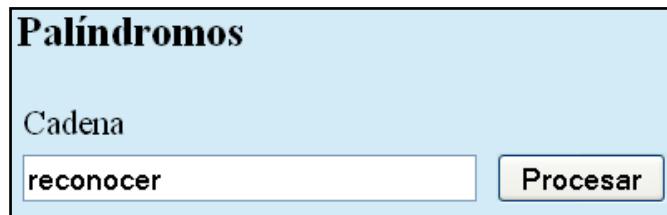
Archivo: prog2\sem03\ejm20.php

```
<?php
$resultado = FALSE;
if( isset($_POST["procesar"]) ) {
    $resultado = TRUE;
    $cad = $_POST["cad"];
    $cadinv = ""; // Cadena invertida
    for( $j = 0; $j < strlen($cad); $j++ ) {
        $cadinv = substr( $cad, $j, 1) . $cadinv;
    }
    if( $cad == $cadinv ) {
        $msg = "Se trata de un palíndromo.";
    } else {
        $msg = "No es un palíndromo";
    }
}
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <h3>Palíndromos</h3>
    <form method="post" action="ejm20.php" accept-charset="utf-8">
        <table width="315">
            <tr>
                <td width="209">Cadena</td>
                <td width="94">&nbsp;</td>
            </tr>
            <tr>
                <td>
                    <input type="text" name="cad" size="30" maxlength="30">
                </td>
                <td>
                    <input name="procesar" type="submit" id="procesar"
                        value="Procesar">
                </td>
            </tr>
        </table>
```

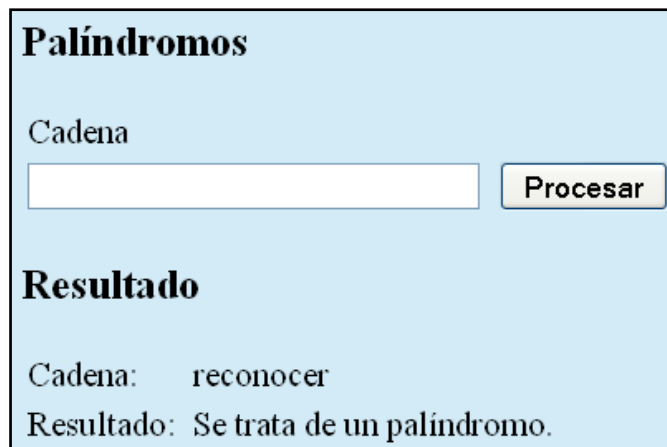
```
</form>
<?php if( $resultado ) { ?>
<div id="resultado">
  <h3>Resultado</h3>
  <table width="500">
    <tr>
      <td width="71">Cadena:</td>
      <td width="417"><?php echo($cad) ?></td>
    </tr>
    <tr>
      <td>Resultado:</td>
      <td><?php echo($msg) ?></td>
    </tr>
  </table>
</div>
<?php } ?>
</body>
</html>
```

A continuación tenemos la interfaz inicial de la ejecución de éste programa:



The image shows a web form titled "Palíndromos". It has a label "Cadena" above a text input field containing the word "reconocer". To the right of the input field is a button labeled "Procesar".

Después de ingresar una cadena y hacer clic sobre el botón **Procesar** obtenemos el siguiente resultado:



The image shows the same web form after processing. The "Cadena" input field is now empty. Below the input field, the text "Resultado" is displayed. Under "Resultado", there are two lines of text: "Cadena: reconocer" and "Resultado: Se trata de un palíndromo." The "Procesar" button remains.

Si queremos analizar otra cadena la ingresamos en el campo respectivo y hacemos clic en el botón **Procesar**.

4 Proyectos Propuestos

4.1 Proyecto 1

Desarrollar un proyecto que permita analizar un número con respecto a las siguientes características:

- Si es primo o no.
- Si es par o impar.
- Si es capicúa o no.

4.2 Proyecto 2

Desarrollar un proyecto que permita calcular el MCD y MCM de dos números.

4.3 Proyecto 3

Desarrollar un proyecto que permita encontrar la suma de los números comprendidos entre n_1 y n_2 , siendo $n_1 < n_2$.

4.4 Proyecto 4

La empresa Luz del Sur necesita de un programa que permita a sus clientes estimar el importe de su recibo de luz.

El costo de un KW/h está en función a la cantidad que consume, si más KW/h se consume el costo será mayor, según el siguiente cuadro:

Consumo (KW/h)	Costo por KW/h (Soles)
Hasta 500 KW/h	0.70
De 501 a 1000 KW/h	0.85
De 1001 a 1500 KW/h	1.15
De 1501 a 2000 KW/h	1.50
De 2001 a más KW/h	2.50

La cantidad de KW/h que consume un cliente se realiza en función a dos lecturas que la empresa realiza los días 20 de cada mes.