

PROGRAMACIÓN II



SEMANA 04 MANEJO DE LIBRERIAS

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

MANEJO DE LIBRERÍAS

En toda aplicación siempre encontramos segmentos de código que repetimos en varias partes de un programa o en varios programas, en estos casos se recomienda utilizar funciones y librerías para poder reutilizarlas en lugar de estar duplicando el código.

En esta oportunidad, aprenderás a crear funciones y librerías, las librerías nos permiten agrupar funciones y luego incluirlas en diferentes programas.

Índice

1	CREACIÓN DE FUNCIONES	4
2	PARÁMETROS DE LAS FUNCIONES	9
2.1	PARÁMETROS POR VALOR.....	9
2.2	PASAR PARÁMETROS POR REFERENCIA.....	11
2.3	PARÁMETROS POR DEFECTO	12
2.4	NÚMERO VARIABLE DE PARÁMETROS	14
3	DEVOLVIENDO VALORES	17
4	INCLUIR ARCHIVOS	21
4.1	FUNCIONES: REQUIRE() E INCLUDE().....	21
4.2	FUNCIONES: REQUIRE_ONCE() E INCLUDE_ONCE().....	21
5	USO DE LIBRERÍAS	23

1 Creación de Funciones

Una función no es más que un bloque de código bajo un identificador al que le pasamos una serie de parámetros y nos devuelve un valor. Como todos los lenguajes de programación, PHP tiene implementado una gran cantidad de funciones para nuestro uso, pero las funciones más importantes son las que nosotros creamos. Para definir una función se debe utilizar la siguiente sintaxis:

```
function nombre_funcion ( $parm_1, $parm_2, ..., $parm_n ) {  
  
    // Cuerpo de la función  
  
}
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función.

Ejemplo 1

Este ejemplo ilustra el uso de una función que no retorna ningún valor, solo ejecuta una acción.

Archivo: prog2\sem04\ejm01.php

```
<?php  
function say( $cad ) {  
    echo $cad . "\n" ;  
}  
?>  
  
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  
    <title>Capítulo 07 - Ejemplo 01</title>  
  </head>  
  <body bgcolor="#D2EBF7">  
    <h3>Aplicando Funciones</h3>  
    <p><?php say("Hola Mundo") ?></p>  
  </body>  
</html>
```

Resultado:

Aplicando Funciones

Hola Mundo

Hasta PHP 3, las funciones deben definirse antes de que sean referenciadas. A partir de PHP 4 no existe tal requerimiento. Excepto cuando una función es definida condicionalmente.

Ejemplo 2

En este ejemplo podemos apreciar cómo podemos utilizar las funciones dentro de un programa.

Archivo: prog2\sem04\ejm02.php

```
<?php

$nota = rand(0,20);    // Genera una nota

if( $nota >= 14 ) {
    function fn_msg() {
        say( "<h4>Felicitaciones!!!</h4>" );
    }
} else {
    function fn_msg() {
        say( "<h4>Intentalo en otra oportunidad!!!</h4>" );
    }
}

// *****
// Funciones Globales
// *****

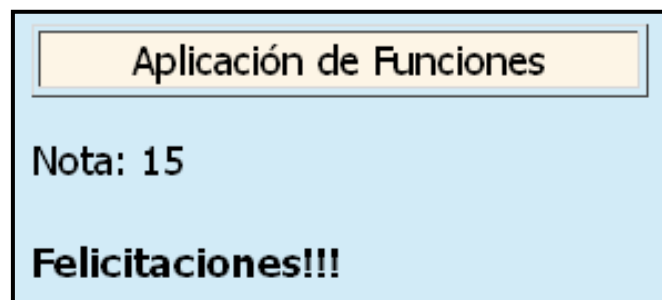
function say( $cad ) {
    echo( $cad . "\n" );
}

function fn_titulo() {
    say( "<table border='1' width='250'>" );
    say( "<tr>" );
    say( "<td align='center' valign='middle' bgcolor='#FDF5E6'>" );
    say( "Aplicación de Funciones" );
    say( "</td>" );
    say( "</tr>" );
}
```

```
say( "</table>" );
}
?>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 07 - Ejemplo 02</title>
  </head>
  <body bgcolor="#D2EBF7" style="font-family: Tahoma;">
    <?php
      fn_titulo();
      say( "<p>Nota: $nota</p>" );
      fn_msg();
    ?>
  </body>
</html>
```

Resultado:



Las funciones también pueden retornar un valor, para lo cual debemos utilizar la instrucción **return**.

Ejemplo 3

En este ejemplo se ilustra el uso de la instrucción **return** para hacer que una función retorne un valor, en este caso la función **fn_factorial** retorna el factorial de **\$n**.

Archivo: prog2\sem04\ ejm03.php

```
<?php

function say( $cad ) {
  echo( $cad . "\n" );
}
```

```
function fn_factorial( $n ) {
    $f = 1;
    for( $j = 1; $j <= $n; $j++ ){
        $f *= $j;
    }
    return $f;
}

?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 07 - Ejemplo 03</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
    <h3>Factorial</h3>
    <table border='1' width='250'>
        <tr>
            <td align='center' valign='middle' bgcolor='#FDF5E6'>
                Número
            </td>
            <td align='center' valign='middle' bgcolor='#FDF5E6'>
                Factorial
            </td>
        </tr>
        <?php for( $i = 1; $i <= 5; $i++ ) { ?>
            <tr>
                <td align='center' valign='middle' bgcolor='#FDF5E6'>
                    <?php say( $i ) ?>
                </td>
                <td align='center' valign='middle' bgcolor='#FDF5E6'>
                    <?php say( fn_factorial($i) ); ?>
                </td>
            </tr>
        <?php } ?>
    </table>
</body>
</html>
```

Resultado:

Factorial

Número	Factorial
1	1
2	2
3	6
4	24
5	120

2 Parámetros de las funciones

La información que se le envía y recibe de una función normalmente es a través de parámetros. Los parámetros se convierten automáticamente en variables cuyo alcance es el contexto de la función.

Las funciones PHP pueden tener cuatro tipos de parámetros:

1. Parámetros por valor
2. Parámetros por referencia
3. Parámetros por defecto
4. Número variable de parámetros

2.1 Parámetros por Valor

Este tipo de parámetro se utiliza para suministrar a las funciones una serie de valores al momento de invocarlas, estos valores se denominan "**parámetros actuales**" y los parámetros (variables) que reciben estos valores se denominan "**parámetros formales**", este es el comportamiento por defecto de los parámetros definidos en una función.

Ejemplo 4

En el presente ejemplo se ha creado la función **fn_promedio()** para calcular el promedio de dos números, esta función tiene dos parámetros formales: \$n1 y \$n2, estos parámetros reciben el valor de los parámetros actuales: \$nota1 y \$nota2, luego calcula el promedio y lo retorna el resultado.

Archivo: prog2\sem04\ejm04.php

```
<?php

function fn_promedio( $n1, $n2 ) {
    $pr = ( $n1 + $n2 ) / 2;
    return $pr;
}

$nota1 = rand(0, 20);
$nota2 = rand(0,20);
$prom  = fn_promedio( $nota1, $nota2 );
$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];

?>
```



```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Capítulo 07 - Ejemplo 04</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
  <table border='1' width='250'>
    <tr>
      <td colspan="2" align='center' valign='middle' bgcolor='#FDF5E6'>
        Calculo de Promedio
      </td>
    </tr>
    <tr>
      <td width="116">Nota 1</td>
      <td width="118"><?php echo($nota1) ?></td>
    </tr>
    <tr>
      <td>Nota 2</td>
      <td><?php echo($nota2) ?></td>
    </tr>
    <tr>
      <td>Promedio</td>
      <td><?php echo($prom) ?></td>
    </tr>
  </table>
  <a href="<?php echo($pagina) ?>">Nueva Prueba</a>
</body>
</html>
```

Resultado

Calculo de Promedio	
Nota 1	20
Nota 2	15
Promedio	17.5
Nueva Prueba	

2.2 Pasar Parámetros por Referencia

Por defecto, los parámetros de una función se pasan por valor, de manera que si cambiamos el valor del parámetro formal dentro de la función, el parámetro actual no es afectado, esto quiere decir que lo que se transfiere del parámetro actual al parámetro formal es una copia del dato.

Si requerimos que una función pueda modificar los parámetros actuales, estos deben ser pasados por referencia, lo que quiere decir que se transfiere la dirección de memoria del parámetro actual.

Para que un parámetro sea pasado por referencia debemos anteponer un ampersand (&) al nombre del parámetro formal en la definición de la función.

Ejemplo 5

En este ejemplo se ilustra el uso de un parámetro por referencia en la función **fn_add()**, el parámetro por referencia es **\$n**, cuyo valor es incrementado en el valor que recibe **\$inc**.

Archivo: prog2\sem04\ ejm05.php

```
<?php
function fn_add( &$n, $inc ) {
    $n += $inc;
}

$numGen = rand( 20, 50 );
$delta = rand( 20, 50 );

$resultado = $numGen;
fn_add($resultado, $delta);

$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 07 - Ejemplo 05</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
    <table border='1' width='272'>
        <tr>
            <td colspan="2" align='center' bgcolor='#FDF5E6'>
                <b>Parámetro por Referencia</b>
            </td>
        </tr>
    </table>
</body>
</html>
```

```

        </td>
    </tr>
    <tr>
        <td width="137">Número</td>
        <td width="119"><?php echo($numGen) ?></td>
    </tr>
    <tr>
        <td>Incremento</td>
        <td><?php echo($delta) ?></td>
    </tr>
    <tr>
        <td>Nuevo Número</td>
        <td><?php echo($resultado) ?></td>
    </tr>
</table>
<br/>
<a href="<?php echo($pagina) ?>">Nueva Prueba</a>
</body>
</html>

```

Resultado:

Parámetro por Referencia	
Número	41
Incremento	42
Nuevo Número	83

[Nueva Prueba](#)

2.3 Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++.

El valor por defecto tiene que ser una expresión constante. Cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto; de otra manera la función no se ejecutará de la forma esperada.

Ejemplo 6

En el presente ejemplo se ha modificado el parámetro **\$inc** en la función **fn_add()** para que tenga como valor por defecto uno (1), quiere decir que si no especificamos valor para este parámetro, tomará valor 1.

Archivo: prog2\sem04\ ejm06.php

```
<?php
function fn_add( &$n, $inc = 1 ) {
    $n += $inc;
}

$numGen = rand( 20, 50 );
$resultado = $numGen;
fn_add($resultado);

$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 07 - Ejemplo 06</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
    <table border='1' width='272'>
        <tr>
            <td colspan="2" align='center' bgcolor='#FDF5E6'>
                <b>Parámetro por Defecto</b>
            </td>
        </tr>
        <tr>
            <td width="137">Número</td>
            <td width="119"><?php echo($numGen) ?></td>
        </tr>
        <tr>
            <td>Incremento</td>
            <td>1</td>
        </tr>
        <tr>
            <td>Nuevo Número</td>
            <td><?php echo($resultado) ?></td>
        </tr>
    </table>
    <br/>
    <a href="<?php echo($pagina) ?>">Nueva Prueba</a>
</body>
</html>
```

Resultado:

Parámetro por Defecto	
Número	41
Incremento	1
Nuevo Número	42
Nueva Prueba	

2.4 Número Variable de Parámetros

A partir de PHP 4 las funciones definidas por el usuario soportan listas de longitud variable de parámetros. La manipulación de estos parámetros se realiza con las siguientes funciones, que solo pueden ser invocadas dentro de una función:

Función	Descripción
func_num_args()	Devuelve el número de parámetros pasados a la función actual definida por el usuario.
func_get_arg(n)	Devuelve el valor de un parámetro específico, n representa la posición del parámetro, e inicia en cero (0).
func_get_args()	Devuelve un array que contiene la lista de parámetros pasados a la función.

Ejemplo 7

En este ejemplo se ilustra como pasar y procesar un número variable de parámetros. La función **fn_suma()** puede recibir varios parámetros, luego internamente realiza la suma de todos ellos y retorna el resultado.

Archivo: prog2\sem04\ejm07.php

```
<?php
function fn_suma() {
    $nums = func_get_args();
    $suma = 0;
    foreach( $nums as $valor ) {
        $suma += (double)$valor;
    }
}
```

```
}
return $suma;
}

$lista = array( rand(20,50), rand(20,50), rand(20,50) );
$suma = fn_suma( $lista[0], $lista[1], $lista[2] );

$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Capítulo 07 - Ejemplo 08</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
<table border='1' width='250'>
<tr>
<td colspan="2" align='center' valign='middle' bgcolor='#FDF5E6'>
<b>Parámetros Variables</b>
</td>
</tr>
<tr>
<td width="88">Números</td>
<td width="146">
<?php
$numeros = "";
foreach ($lista as $dato){
$numeros .= "$dato ";
}
echo($numeros);
?>
</td>
</tr>
<tr>
<td>Suma</td>
<td><?php echo($suma) ?></td>
</tr>
</table>
<br>
<a href="<?php echo($pagina) ?>">Nueva Prueba</a>
</body>
</html>
```

Resultado:

Parámetros Variables	
Números	21 27 28
Suma	76
Nueva Prueba	

3 Devolviendo valores

Las funciones pueden retornar valores, para eso se debe utilizar la instrucción `return`, su sintaxis es la siguiente:

```
return valor_de_retorno;
```

El **valor_de_retorno** puede ser una constante, la llamada a otra función ó una expresión.

Ejemplo 8

En este ejemplo se ilustra el uso de la instrucción **return**, la función **fn_mayor()** recibe tres números enteros y retorna el mayor de ellos.

Archivo: prog2\sem04\ ejm08.php

```
<?php

function fn_mayor( $n1, $n2, $n3 ) {
    $mayor = $n1;
    if( $n2 > $mayor ) {
        $mayor = $n2;
    }
    if( $n3 > $mayor ) {
        $mayor = $n3;
    }
    return $mayor;
}

$a = rand(1,100);
$b = rand(1,100);
$c = rand(1,100);
$m = fn_mayor( $a, $b, $c );

$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 07 - Ejemplo 08</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
```



```

<table border='1' width='201'>
  <tr>
    <td colspan="2" align='center' valign='middle' bgcolor='#FDF5E6'>
      <b>Número Mayor</b>
    </td>
  </tr>
  <tr>
    <td width="90">Nro. 1</td>
    <td width="95"><?php echo($a) ?></td>
  </tr>
  <tr>
    <td>Nro. 2</td>
    <td><?php echo($b) ?></td>
  </tr>
  <tr>
    <td>Nro. 3</td>
    <td><?php echo($c) ?></td>
  </tr>
  <tr>
    <td>Mayor</td>
    <td><?php echo($m) ?></td>
  </tr>
</table>
<a href="<?php echo($pagina) ?>">Nueva Prueba</a>
</body>
</html>

```

Resultado:

Número Mayor	
Nro. 1	74
Nro. 2	13
Nro. 3	94
Mayor	94
Nueva Prueba	

Si queremos que una función retorne varios valores, esto es posible haciendo que retorne un arreglo.

Ejemplo 9

En este ejemplo se ha creado la función **fn_array()** que retorna un arreglo cuyo tamaño se lo pasamos a través del parámetro **\$n** y los elementos se generan en forma aleatoria.

Archivo: prog2\sem04\ ejm09.php

```
<?php

function fn_array( $n ) {
    $datos = array();
    for( $j = 1; $j <= $n; $j++ ){
        $datos[] = rand(1,100);
    }
    return $datos;
}

$size = rand(5,15); // Tamaño del arreglo
$list = fn_array($size); // Genera el arreglo

$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 07 - Ejemplo 09</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">

    <table border='1' width='392'>
        <tr>
            <td colspan="2" align='center' valign='middle' bgcolor='#FDF5E6'>
                <b>Arreglo Dinámico</b>
            </td>
        </tr>
        <tr>
            <td width="102">Tamaño:</td>
            <td width="274"><?php echo($size) ?></td>
        </tr>
        <tr>
            <td>Valores:</td>
            <td>
                <?php
                foreach ($list as $value) {
                    echo("$value ");
                }
            </td>
        </tr>
    </table>

```

```
}  
?>  
</td>  
</tr>  
</table>  
<a href="<?php echo($pagina) ?>">Nueva Prueba</a>  
</body>  
</html>
```

Resultado:

Arreglo Dinámico	
Tamaño:	11
Valores:	14 82 81 52 39 62 23 87 67 62 73
Nueva Prueba	

4 Incluir Archivos

4.1 Funciones: `require()` e `include()`

El objetivo de estas dos funciones es incluir el contenido de un archivo en el punto donde se invoca. En su funcionamiento son idénticas en todos los aspectos excepto en el modo de actuar ante un error; **`include()`** produce un mensaje de advertencia, mientras que **`require()`** produce un Error Fatal.

4.2 Funciones: `require_once()` e `include_once()`

Estas funciones son similares a sus pares **`require()`** e **`include()`** respectivamente, con la diferencia que si el archivo ya fue incluido no se volverá a incluir.

Ejemplo 10

En este ejemplo se ilustra el uso de la función **`require()`**, para lo cual utilizamos los siguientes archivos:

Archivo	Descripción
titulo.html	Archivo que contiene las instrucciones de cabecera de página.
ejm10.php	Programa donde se incluirá el archivo titulo.html .

Archivo: `prog2\sem04\titulo.html`

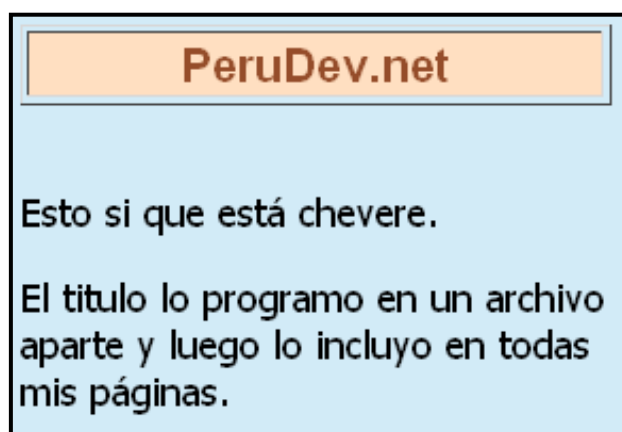
```
<style>
  .titulo{
    background-color: #FFDFC1;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: bold;
    font-size: 20px;
    text-align: center;
    vertical-align: middle;
    color: #954D2A;
  }
</style>
<table border='1' width='250'>
  <tr>
    <td class="titulo">
```

```
        PeruDev.net
    </td>
</tr>
</table>
<br>
```

Archivo: prog2\sem04\ejm10.php

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Ejemplo 10</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
  <?php require( "titulo.html" ); ?>
  <p>Esto si que está chevere.</p>
  <p>
    El titulo lo programo en un archivo<br />
    aparte y luego lo incluyo en todas<br />
    mis páginas.
  </p>
</body>
</html>
```

Cuando ejecutamos el programa **ejm10.php** en el navegador el resultado que obtenemos es:



5 Uso de Librerías

La aplicación más interesante de las funciones **require()** e **include()** es con la inclusión de archivos que contienen un conjunto de funciones, esto nos permite tener librerías de rutinas comunes que puedan ser reutilizadas en cualquier aplicación.

Ejemplo 11

En el presente ejemplo se ilustra el uso de una librería de rutinas, los archivos a utilizar son:

Archivo	Descripción
cabecera.html	Archivo que contiene la sección de título de la página.
pie.html	Archivo que contiene la sección de pie de página.
libreria.php	Archivo que contiene funciones comunes que pueden ser compartidas por múltiples programas.
estilo.css	Archivo que contiene los estilos de la página.
ejm0811.php	Programa donde se incluirán los archivos anteriores.

Archivo: prog2\sem04\cabecera.html

```
<table width="550">
  <tr>
    <td width="150"></td>
    <td width="381" align="center">
      <h1>PeruDev</h1>
      <h4>Cursos, Libros, Programas, y mucho mas.</h4>
    </td>
  </tr>
</table>
```

Archivo: prog2\sem04\pie.html

```
<br/>
<table width="550">
  <tr>
    <td width="542" align="center">
      Copyright @ 2009 PeruDev - Todos los derechos reservados <br />
      Lima - Perú
```

```
</td>
</tr>
</table>
```

Archivo: prog2\sem04\libreria.php

```
<?php

function fn_VerificarTriangulo( $a, $b, $c ) {
    if( $a > abs($b - $c) and $a < ($b + $c) ) {
        $ok1 = TRUE;
    } else {
        $ok1 = FALSE;
    }
    if( $b > abs($a - $c) and $b < ($a + $c) ) {
        $ok2 = TRUE;
    } else {
        $ok2 = FALSE;
    }
    if( $c > abs($a - $b) and $c < ($a + $b) ) {
        $ok3 = TRUE;
    } else {
        $ok3 = FALSE;
    }
    if( $ok1 and $ok2 and $ok3 ) {
        return TRUE;
    } else {
        Return FALSE;
    }
}

function fn_CalcularAreaTriangulo( $a, $b, $c ) {
    $p = ( $a + $b + $c ) / 2;
    $area = sqrt( $p * ($p - $a) * ($p - $b) * ($p - $c) );
    return $area;
}

?>
```

Archivo: prog2\sem04\estilo.css

```
BODY {
    color: #000099;
```

```
background-color: #D2EBF7;
font-family:Verdana, Arial, Helvetica, sans-serif;
font-size: 0.8em;
font-weight: 500;
}

H1{
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 2em;
font-weight: 700;
}

H2{
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 1.75em;
font-weight: 700;
}

H3{
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 1.58em;
font-weight: 500;
}

TD {
font-family: Verdana, Arial, Helvetica, sans-serif;
font-size: 0.7em;
font-weight: 500;
}
```

Archivo: prog2\sem04\ejm11.php

```
<?php
require( "libreria.php" );

$formulario = TRUE;

if( isset($_POST["calcular"]) ) {
    $formulario = FALSE;
    $a = $_POST["a"];
    $b = $_POST["b"];
    $c = $_POST["c"];
    if( fn_VerificarTriangulo( $a, $b, $c ) ) {
        $area = fn_CalcularAreaTriangulo( $a, $b, $c );
        $msg = "Proceso Ok.";
    }
}
```



```

    } else {
        $area = "Error";
        $msg = "No existe el triangulo.";
    }
}

$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Capítulo 07 - Ejemplo 11</title>
    <link rel="stylesheet" href="estilo.css" type="text/css"/>
</head>
<body>
    <?php require( "cabecera.html" ); ?>
    <h2>Calculo del Area de un Triangulo</h2>
    <?php if( $formulario ) { ?>
    <h3>Ingrese el Valor de los Lados</h3>
    <form action="<?php echo $pagina; ?>" method="post">
        <table width="156">
            <tr>
                <td width="87">Lado A</td>
                <td width="57">
                    <input type="text" name="a" size="5" maxlength="3">
                </td>
            </tr>
            <tr>
                <td>Lado B</td>
                <td><input type="text" name="b" size="5" maxlength="3"></td>
            </tr>
            <tr>
                <td>Lado C</td>
                <td><input type="text" name="c" size="5" maxlength="3"></td>
            </tr>
        </table>
        <input type="submit" value="Enviar" name="calcular">
        <input type="reset" value="Limpiar">
    </form>
    <?php } else { ?>
    <h3>Resultado del Proceso</h3>
    <table width="429">
        <tr>
            <td width="87">Lado A</td>
            <td width="330"><?php echo($a) ?></td>
        </tr>
    </table>

```

```
<tr>
  <td>Lado B</td>
  <td><?php echo($b) ?></td>
</tr>
<tr>
  <td>Lado C</td>
  <td><?php echo($c) ?></td>
</tr>
<tr>
  <td>Área</td>
  <td><?php echo($area) ?></td>
</tr>
<tr>
  <td>Mensaje</td>
  <td><?php echo($msg) ?></td>
</tr>
</table>
<a href="<?php echo($pagina) ?>">Otra Prueba</a>
<?php } ?>
<?php require( "pie.html" ); ?>
</body>
```

Cuando ejecutamos el programa ejm11.php, inicialmente tenemos el formulario para ingresar los lados de un triángulo, tal como se muestra a continuación:



PeruDev
Cursos, Libros, Programas, y mucho mas.

Calculo del Area de un Triangulo

Ingresa el Valor de los Lados


Lado A

Lado B

Lado C

Copyright © 2009 PeruDev - Todos los derechos reservados
Lima - Perú

Cuando hacemos clic en el botón **Enviar** tenemos el siguiente resultado:

**PeruDev**
Cursos, Libros, Programas, y mucho mas.

Calculo del Area de un Triangulo

Resultado del Proceso

Lado A	3
Lado B	4
Lado C	5
Área	6
Mensaje	Proceso Ok.
Otra Prueba	

Copyright © 2009 PeruDev - Todos los derechos reservados
Lima - Perú

Como podemos apreciar el uso de librerías nos facilita la reutilización de código mediante funciones.