



TALLER DE PROGRAMACIÓN WEB



SEPARATA IV: PROGRAMANDO CON PHP

Eric Gustavo Coronel Castillo
gcoronel@uni.edu.pe
gcoronelc.github.io

ÍNDICE

CAPÍTULO 1 COMENZANDO CON PHP	7
INSERTANDO CÓDIGO PHP	7
<i>Caso 1</i>	7
<i>Caso 2</i>	9
<i>Caso 3</i>	10
<i>Caso 4</i>	11
INSTRUCCIONES BÁSICAS.....	12
<i>Separación de Instrucciones</i>	12
<i>Combinando PHP y HTML</i>	13
<i>Comentarios</i>	15
<i>Variables</i>	17
TIPOS DE DATOS	19
<i>Enteros</i>	19
<i>Números de Punto Flotante</i>	20
<i>Cadenas</i>	21
<i>Averiguar el Tipo de una Variable</i>	26
<i>Valor Lógico (Boolean)</i>	27
<i>Forzado de tipos</i>	28
IMPRESIÓN EN EL NAVEGADOR	31
<i>Instrucción: echo</i>	31
<i>Función: print()</i>	32
<i>Función: printf()</i>	33
<i>Constantes</i>	35
EXPRESIONES Y OPERADORES	39
<i>Operadores Aritméticos</i>	39
<i>Operadores de Asignación</i>	39
<i>Operadores de Comparación</i>	41
<i>Operador de Ejecución</i>	42
<i>Operadores de Incremento/Decremento</i>	43

<i>Operadores Lógicos</i>	44
<i>Operadores de Cadena</i>	46
PRÁCTICA DE LABORATORIO	48
<i>Revisar la instalación de PHP</i>	48
<i>Generar Tabla de Multiplicar</i>	48
<i>División de Dos Números</i>	48
CAPÍTULO 2 VARIABLES EXTERNAS	49
FORMULARIOS HTML (GET Y POST).....	49
DIRECTIVA REGISTER_GLOBALS	51
ARREGLOS GLOBALES HTTP	53
<i>Arreglo \$_POST</i>	53
<i>Arreglo \$_GET</i>	56
PROGRAMAS RECURSIVOS.....	59
COOKIES HTTP	62
PROYECTOS PROPUESTOS	64
<i>Proyecto 1</i>	64
CAPÍTULO 3 ESTRUCTURAS DE CONTROL	65
INTRODUCCIÓN	65
ESTRUCTURAS CONDICIONALES	66
<i>Condición Simple: if</i>	66
<i>Condición Doble: if – else</i>	69
<i>Condición Múltiple: if – elseif – else</i>	72
<i>Selectiva Múltiple: switch</i>	76
ESTRUCTURAS REPETITIVAS.....	82
<i>Bucle: while</i>	82
<i>Bucle: do – while</i>	87
<i>Bucle: for</i>	89
<i>Bucle: foreach</i>	92
<i>Instrucciones: break y continue</i>	95
PROYECTOS PROPUESTOS	109
<i>Proyecto 1</i>	109

<i>Proyecto 2</i>	109
<i>Proyecto 3</i>	109
<i>Proyecto 4</i>	109
CAPÍTULO 4 MANEJO DE LIBRERÍAS	110
CREACIÓN DE FUNCIONES	110
PARÁMETROS DE LAS FUNCIONES	115
<i>Parámetros por Valor</i>	115
<i>Pasar Parámetros por Referencia</i>	117
<i>Parámetros por defecto</i>	119
<i>Número Variable de Parámetros</i>	121
DEVOLVIENDO VALORES.....	123
INCLUIR ARCHIVOS	127
<i>Funciones: require() e include()</i>	127
<i>Funciones: require_once() e include_once()</i>	127
USO DE LIBRERÍAS	129
CAPÍTULO 5 ARREGLOS.....	135
¿QUÉ ES UN ARREGLO?.....	135
ARREGLOS UNIDIMENSIONALES	137
ARREGLOS MULTIDIMENSIONALES.....	142
CAPÍTULO 6 ARREGLOS ASOCIATIVOS	145
DEFINICIÓN	145
USO DE LA FUNCIÓN ARRAY()	147
USO DE FOREACH()	149
CAPÍTULO 7 SESIONES	151
¿QUÉ SON LAS SESIONES?.....	151
MANEJO DE SESIONES	151
<i>Función: sesión_start()</i>	151
<i>Función: session_id()</i>	153
<i>Arreglo: \$_SESSION</i>	154
<i>Función: session_unset()</i>	156
<i>Función: session_destroy()</i>	156

EJEMPLO APLICATIVO	160
--------------------------	-----

Capítulo 1

COMENZANDO CON PHP

INSERTANDO CÓDIGO PHP

Los scripts basados en PHP están insertados en el código HTML, y para esto tenemos tres formas, que se describen a continuación.

Caso 1

Esta primera forma sólo está disponible si se han habilitado las etiquetas cortas. Esto se puede hacer habilitando la directiva de configuración `short_open_tag` en el archivo de configuración de PHP.

Sintaxis

```
<?

// Aquí se inserta el script PHP

?>
```

Por defecto la directiva `short_open_tag` está deshabilitada. Si desea usar PHP en conjunto con XML se recomienda que se mantenga deshabilitada, de modo que pueda usar `<?xml ?>` en forma directa.

- Valor por defecto de la directiva `short_open_tag`:

```
short_open_tag = Off
```

- Para habilitarla debe establecerla a `On`:

```
short_open_tag = On
```

Esta directiva afecta también shorthand `<?=expression?>`, la cual es idéntica a `<? echo(expression) ?>`. El uso de este atajo requiere que `short_open_tag` se encuentre habilitado.

Nota:

En general, se recomienda que la directiva **short_open_tag** se mantenga deshabilitada; además, muchos servicios de hosting la mantienen deshabilitada.

Ejemplo 1:

Este ejemplo ilustra el uso de etiquetas cortas para insertar código PHP, recuerde que debe habilitar la directiva `short_open_tag`.

La sentencia `header(...)` es para configurar la salida en el navegador en formato **UTF-8** para que se puedan apreciar los acentos.

Archivo: `cap1/ejm01.php`

```
<?
  header('Content-Type: text/html; charset=utf-8' );
  echo "<h1>Conoce el Perú</h1>";
  echo "<h1>Visita Cusco</h1>";

  $destino2 = "Chiclayo";
?>
<h1>También visita <?=$destino2?></h1>
```

Para probar el ejemplo, en el campo dirección del navegador anote la siguiente URL:

`http://localhost/cap1/ejm01.php`

El resultado es el siguiente:

Conoce el Perú

Visita Cusco

También visita Chiclayo

Caso 2

También denominadas etiquetas largas, es la forma estándar de insertar código PHP, no es necesario hacer configuración alguna para que funcione, cualquier instalación de PHP la soporta.

Sintaxis

```
<?php  
  
    // Aquí se inserta el script PHP  
  
?>
```

Ejemplo 2

Este ejemplo ilustra el uso de etiquetas largas para insertar código PHP.

Archivo: cap1\ejm02.php

```
<?php  
    header('Content-Type: text/html; charset=utf-8' );  
    echo "Apache – PHP – MySQL<br>";  
    echo "Una gran alternativa<br>";  
    $msg = "para hacer Grandes Sistemas.";  
?>  
<?php echo($msg) ?>
```

Para probar el ejemplo, en el campo dirección del navegador anote la siguiente URL:

<http://localhost/cap1/ejm02.php>

El resultado es el siguiente:

Apache – PHP – MySQL Una gran alternativa para hacer Grandes Sistemas.
--

Caso 3

La etiqueta `script` también permite insertar código PHP, pero tiene el inconveniente que no es muy práctica, por lo tanto, su uso en aplicaciones reales es prácticamente nulo, solo se menciona por cultura informática.

Sintaxis

```
<script language="php">  
  
    // Aquí se inserta el script PHP  
  
</script>
```

Ejemplo 3

Este ejemplo ilustra el uso de la etiqueta `script` para insertar código PHP.

Archivo: cap1\ejm03.php

```
<script language="PHP">  
    header('Content-Type: text/html; charset=utf-8' );  
    echo( "Los IDE de Java son una gran<br>");  
    echo("alternativa para desarrollar<br>");  
    echo("proyectos con PHP.");  
</script>
```

Para probar el ejemplo, en el campo dirección del navegador anote la siguiente URL:

<http://localhost/cap1/ejm03.php>

El resultado es el siguiente:

Los IDE de Java son una gran
alternativa para desarrollar
proyectos con PHP.

Caso 4

Para los que todavía recuerdan ASP 3.0, esta forma sólo está disponible si se han habilitado las etiquetas tipo ASP. Esto se puede hacer habilitando la directiva de configuración `asp_tags` en el archivo de configuración de PHP.

Sintaxis

```
<%  
  
    // Aquí se inserta el script PHP  
  
%>
```

Ejemplo 4

Este ejemplo ilustra el uso de etiquetas tipo ASP, recuerde que debe habilitar la directiva `asp_tags`.

Archivo: `cap1\ejm04.php`

```
<%  
    header('Content-Type: text/html; charset=utf-8' );  
    echo("Podemos también usar etiquetas tipo ASP.<br>");  
    echo("No se recomienda su uso.");  
%>
```

Para probar el ejemplo, en el campo dirección del navegador anote la siguiente URL:

`http://localhost/cap1/ejm04.php`

El resultado es el siguiente:

Podemos también usar etiquetas tipo ASP. No se recomienda su uso.
--

INSTRUCCIONES BÁSICAS

Separación de Instrucciones

Las instrucciones se separan igual que en C o PERL, terminando cada sentencia con un punto y coma.

La etiqueta de cierre (`?>`) también implica el fin de la sentencia, así tenemos que el Ejemplo 5 es equivalente al Ejemplo 6.

Ejemplo 5

Archivo: `cap1\ejm05.php`

```
<?php
  header('Content-Type: text/html; charset=utf-8' );
  echo("Que fácil es PHP.<br>");
  echo("Pronto seré un experto.");
?>
```

Ejemplo 6

Archivo: `cap1\ejm06.php`

```
<?php
  header('Content-Type: text/html; charset=utf-8' );
  echo("Que fácil es PHP.<br>");
  echo("Pronto seré un experto.")
?>
```

En ambos ejemplos el resultado es:

Que fácil es PHP.
Pronto seré un experto.

Combinando PHP y HTML

Es posible combinar el código PHP y HTML para producir bloques HTML que serán parte de un condicional o un bucle, tal como se ilustra en los siguientes ejemplos.

Ejemplo 7

Archivo: cap1\ejm07.php

```
<?php
header('Content-Type: text/html; charset=utf-8' );
srand((double)microtime()*1000000);
$nota = rand(0,20);
echo("<h1>Nota: $nota</h1>");
?>
<?php if($nota<14) { ?>
    <h1>Estas Desaprobado.</h1>
<?php } else { ?>
    <h1>Felicitaciones Aprobaste.</h1>
<?php } ?>
```

En este ejemplo el HTML está dentro de un condicional, por lo tanto su ejecución depende en este caso del valor que toma la variable **\$nota**. Su resultado podría ser así:

Nota: 18 Felicitaciones Aprobaste.

Ejemplo 8

Archivo: cap1\ejm08.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<table width="200" border="1">
  <tr>
    <th align="center">Número</th>
    <th align="center">Cuadrado</th>
  </tr>
  <?php for($k=1;$k<=5;$k++){ ?>
    <tr>
      <td align="center"><?php echo( $k ); ?></td>
      <td align="center"><?php echo( $k * $k ); ?></td>
    </tr>
  <?php } ?>
</table>
```

En este ejemplo el código HTML está dentro de un bucle PHP, por lo tanto se repite tantas veces como ejecuciones tenga el bucle; además dentro del código HTML tenemos también código PHP. Note que la segunda fila de la tabla es la que se encuentra dentro del bucle. El resultado es el siguiente:

Número	Cuadrado
1	1
2	4
3	9
4	16
5	25

Comentarios

PHP soporta comentarios tipo C, C++ y Shell de Unix.

Sintaxis 1

Comentario multilínea:

```
/*  
    Inicio del comentario  
    El comentario continúa en esta línea  
    El comentario termina en esta línea  
*/
```

Sintaxis 2

Comentario en línea – Caso 1

```
Sentencia; // Comentario en línea
```

Sintaxis 3

Comentario en línea – Caso 2

```
Sentencia; # Comentario en línea tipo Shell de Unix
```

Ejemplo 9

Archivo: cap1\ejm09.php

```
<?php
/*
 * Este ejemplo ilustra el uso de Comentarios
 * Como podemos observar son tres los tipos
 */
header('Content-Type: text/html; charset=utf-8' );
echo("Ejemplos de Comentarios<br>");
echo("PHP is Powerfull Campeón<br>"); // Mensaje Ganador
echo("Perú Campeón"); # Esperanza de todos los peruanos
?>
```

El resultado es el siguiente:

Ejemplos de Comentarios PHP is Powerfull Campeón Perú Campeón

Variables

Las variables en PHP no necesitan ser declaradas, podemos decir que en PHP las variables son débilmente tipadas.

Toda variable debe tener un nombre al que se le debe anteponer el símbolo \$, además el nombre debe empezar con una letra.

El ámbito de una variable es global a nivel del archivo actual y los archivos incluidos; dentro de una función son locales a la función.

Ejemplo 10

Archivo: cap1\ejm10.php

```
<?php
header('Content-Type: text/html; charset=utf-8' );
$nombre = "Gustavo Coronel";
echo( "Mi nombre es: " . $nombre );
?>
```

En este ejemplo estamos creando la variable nombre, y luego estamos imprimiendo su contenido. El resultado es el siguiente:

Mi nombre es: Gustavo Coronel

Ejemplo 11

Archivo: cap1\ejm11.php

```
1 <?php
2     header('Content-Type: text/html; charset=utf-8' );
3     $x = 20; # $x es un entero
4     echo("\$x es de tipo " . gettype($x) . "<br>");
5     $x = "Viva el Perú"; # $x es un cadena
6     echo("\$x es de tipo " . gettype($x) . "<br>");
7 ?>
```

En este ejemplo, la variable \$x cambia de tipo de dato, en la línea 3 es de tipo entero (*integer*) y en la línea 5 es de tipo cadena (*string*). El resultado es el siguiente:

<p>\$x es de tipo integer \$x es de tipo string</p>

TIPOS DE DATOS

Enteros

Los enteros se pueden especificar usando una de las siguientes sintaxis:

```
$a = 4546; # número decimal  
  
$a = -467; # un número negativo  
  
$a = 0352; # número octal (equivalente a 234 decimal)  
  
$a = 0xA5; # número hexadecimal (equivalente a 65 decimal)
```

Ejemplo 12

Archivo: progphpejm12.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>  
<font face="Verdana">  
<?php  
    $a1 = 4546; # número decimal  
    $a2 = -467; # un número negativo  
    $a3 = 0352; # número octal (equivalente a 234 decimal)  
    $a4 = 0xA5; # número hexadecimal (equivalente a 65 decimal)  
    echo("a1 -> " . $a1 . "<br>");  
    echo("a2 -> " . $a2 . "<br>");  
    echo("a3 -> " . $a3 . "<br>");  
    echo("a4 -> " . $a4 . "<br>");  
?>  
</font>
```

El resultado es el siguiente:

a1 -> 4546
a2 -> -467
a3 -> 234
a4 -> 165

Números de Punto Flotante

Los números en punto flotante (*double*) se pueden especificar utilizando cualquiera de las siguientes sintaxis:

```
$a = 15.234;  
  
$a = 1.8e4;
```

Ejemplo 13

Archivo: cap1\ejm13.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>  
<font face="Verdana">  
<?php  
    $a1 = 15.234;  
    $a2 = 1.8e4;  
    echo("a1 -> " . $a1 . "<br>");  
    echo("a2 -> " . $a2);  
?>  
</font>
```

El resultado es el siguiente:

```
a1 -> 15.234  
a2 -> 18000
```

Cadenas

Las cadenas se especifican usando como delimitadores la comilla simple (') o la comilla doble (").

Ejemplo 14

Archivo: cap1\ejm14.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $cad = "\"Esto es una cadena\"";
    echo $cad;
?>
</font>
```

El resultado es el siguiente:

"Esto es una cadena"

El carácter de barra invertida (\) se puede utilizar para indicar caracteres especiales según el siguiente cuadro:

secuencia	significado
\n	Nueva línea
\r	Retorno de carro
\t	Tabulación horizontal
\\	Barra invertida
\\$	Signo del dólar
\"	Comillas dobles

Ejemplo 15

Archivo: cap1\ejm15.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $nombre = "Gustavo Coronel";
    $cargo = "Jefe de Sistemas";
    $salario = 10000.00;
    echo("Nombre: " . $nombre . "<br>");
    echo("Cargo: " . $cargo . "<br>");
    echo("Salario: \$ " . number_format($salario, 2, '.', ''));
?>
</font>
```

El resultado es el siguiente:

Nombre: Gustavo Coronel Cargo: Jefe de Sistemas Salario: \$ 10,000.00

Cuando una cadena está limitada por comilla doble ("), las variables que están dentro de la cadena se expanden, esto quiere decir que son reemplazadas por su valor, tal como se ilustra en el siguiente ejemplo.

Ejemplo 16

Archivo: cap1\ejm16.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Verdana">
<?php
    $a = 5;
    $b = 9;
    $c = $a * $b;
    echo("Variables dentro de<br>");
    echo("una cadena se expanden<br>");
    echo("\$a = $a<br>");
    echo("\$b = $b<br>");
    echo("\$c = $c<br>");
?>
</font>
```

El resultado es el siguiente:

Variables dentro de una cadena se expanden \$a = 5 \$b = 9 \$c = 45

También podemos delimitar las cadenas con comillas simples, pero en este caso las únicas secuencias permitidas son la doble barra invertida (\\) y la comilla simple (\\).

Igualmente, se puede insertar código HTML como parte de la cadena, tal como se ilustra en el siguiente ejemplo.

Ejemplo 17

Archivo: cap1\ejm17.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<?php

    echo("<font face=\"Verdana\">\n");
    echo("<H1>Software Libre</H1>\n");
    echo("<H2>Es la Alternativa</H2>\n");
    echo("</font>");

?>
```

El resultado es el siguiente:

Software Libre

Es la Alternativa

Si revisamos el código HTML que llega al navegador, obtenemos lo siguiente:

```
<font face="Verdana">
<H1>Software Libre</H1>
<H2>Es la Alternativa</H2>
</font>
```

Para concatenar dos cadenas se utiliza el operador punto (.). A continuación, tenemos un ejemplo ilustrativo.

Ejemplo 18

Archivo: cap1\ejm18.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $s1 = "<b>\\"Software Libre\\"</b><br>";
    $s2 = "es una muy buena alternativa<br>";
    $s3 = "para Desarrollar Soluciones<br>";
    $s4 = "Empresariales Seguras y Confiables";
    echo($s1.$s2.$s3.$s4.".");
?>
</font>
```

El resultado es el siguiente:

<p>"Software Libre" es una muy buena alternativa para Desarrollar Soluciones Empresariales Seguras y Confiables.</p>

Averiguar el Tipo de una Variable

Una forma de determinar el tipo de dato de una variable es utilizando la función `gettype()`, tal como se ilustra en el siguiente ejemplo.

Ejemplo 19

Archivo: cap1\ejm19.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php

    $nombre = "Claudia";
    $edad = 25;
    $salario = 3500.00;
    echo("El tipo de \$nombre es: " . gettype($nombre) . "<br>");
    echo("El tipo de \$edad es: " . gettype($edad) . "<br>");
    echo("El tipo de \$salario es: " . gettype($salario));

?>
</font>
```

El resultado es el siguiente:

El tipo de \$nombre es: string El tipo de \$edad es: integer El tipo de \$salario es: double
--

Valor Lógico (Boolean)

Este es el tipo más simple. Un `boolean` expresa un valor de verdad. Puede ser `TRUE` o `FALSE`.

Para especificar un literal `boolean` se usa las palabras reservadas `TRUE` o `FALSE`. Ambas son insensibles a mayúsculas y minúsculas.

Para convertir explícitamente un valor a `boolean`, se debe hacer casting entre tipos de datos usando `bool` o `boolean`. Sin embargo, en la mayoría de casos no es necesario hacer casting, ya que un valor será convertido automáticamente si un operador, función o estructura de control requiere un argumento tipo `boolean`.

Cuando se realizan conversiones a `boolean`, los siguientes valores son considerados `FALSE`:

- El literal boolean `FALSE`
- El integer 0 (cero)
- El float 0.0 (cero)
- El valor string vacío, y el string "0"
- Un array con cero elementos
- Un object con cero variables miembro (sólo en PHP 4)
- El tipo especial `NULL` (incluyendo variables no definidas)
- Objetos SimpleXML creados desde etiquetas vacías

Cualquier otro valor es considerado `TRUE`.

Ejemplo 20

Archivo: cap1\ejm20.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $cad = "<b>Alianza Campeón</b><br>";
    echo($cad);
    if ($cad){
        $estado = "Es verdadero";
    }else{
        $estado = "Es falso";
    }
    echo($estado);
?>
</font>
```

En este ejemplo la variable **\$cad** no está vacía, por lo tanto, se interpreta como verdadera. El resultado es el siguiente:

Alianza Campeón
Es verdadero

Forzado de tipos

El forzado de tipos se conoce como casting y en PHP el funcionamiento es similar a como funciona en C, el nombre del tipo deseado se escribe entre paréntesis antes de la variable a la que se pretende forzar.

```
$a = 10; // $a es un entero
$b = (double) $a; // $b es un doble
```

Los casting permitidos son:

Casting	Descripción
(int), (integer)	Fuerza a entero (integer)
(real), (double), (float)	Fuerza a doble (double)
(string)	Fuerza a cadena (string)
(array)	Fuerza a array (array)
(object)	Fuerza a objeto (object)

Las tabulaciones y espacios se permiten dentro de los paréntesis, así que los siguientes ejemplos son funcionalmente equivalentes:

```
$a = (int) $b;  
$a = ( int ) $b;
```

Puede no ser obvio que ocurrirá cuando se hace casting entre ciertos tipos. Por ejemplo, lo siguiente debería ser tenido en cuenta.

Cuando se hace casting de un escalar o una variable de cadena a un arreglo, la variable se convertirá en el primer elemento del arreglo.

Ejemplo 21

Archivo: cap1\ejm21.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $cad = "Si lo intentas lo lograrás";
    $arr = (array) $cad;
    echo($arr[0]);
?>
</font>
```

El resultado es el siguiente:

Si lo intentas lo lograrás

Cuando se hace casting de una variable escalar o de una cadena a un objeto, la variable se convertirá en un atributo del objeto; el nombre del atributo será *scalar*.

Ejemplo 22

Archivo: cap1\ejm22.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $msg = "<b>iEs libre el que vive según su elección!</b>";
    $obj = (object) $msg;
    echo($obj->scalar);
?>
</font>
```

El resultado es el siguiente:

¡Es libre el que vive según su elección!

IMPRESIÓN EN EL NAVEGADOR

Instrucción: echo

Imprime una o más cadenas. El uso de paréntesis es opcional.

Sintaxis

```
echo (string arg1, string [argn]...)
```

Ejemplo 23

Archivo: cap1\ejm23.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    echo "<b>Hola Mundo</b><br>";
    echo "Esto se extiende
    por varias líneas. <br>También puedes
    insertar código HTML";
?>
</font>
```

El resultado es el siguiente:

<p>Hola Mundo Esto se extiende por varias líneas. También puedes insertar código HTML</p>
--

Función: print()

Imprime una cadena.

Sintaxis

```
print (string arg)
```

Ejemplo 24

Archivo: cap1\ejm24.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    print("<b>Mensaje</b><br>");
    $cad = "El sabio piensa todo lo que dice,<br>";
    $cad = $cad . "pero no dice todo lo que piensa.";
    print($cad);
?>
</font>
```

El resultado es el siguiente:

Mensaje
El sabio piensa todo lo que dice,
pero no dice todo lo que piensa.

Función: printf()

Imprime una cadena con formato.

Sintaxis

```
int printf (string formato [, mixed args...])
```

El formato debe tener el siguiente patrón:

```
%[carácter_de_relleno][ancho][.precisión]tipo
```

Los tipos posibles se especifican en la siguiente tabla:

Tipo	Descripción
b	El argumento es tratado como un entero y presentado como un número binario.
c	El argumento es tratado como un entero, y presentado como el caracter con dicho valor ASCII.
d	El argumento es tratado como un entero y presentado como un número decimal.
f	El argumento es tratado como un doble y presentado como un número de coma flotante
o	El argumento es tratado como un entero, y presentado como un número octal.
s	El argumento es tratado como una cadena y es presentado como tal.
x	El argumento es tratado como un entero y presentado como un número hexadecimal (con minúsculas).
X	El argumento es tratado como un entero y presentado como un número hexadecimal (con mayúsculas).

Ejemplo 25

Archivo: cap1\ejm25.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $A = 10;
    $B = 15.5;
    $C = "MySQL la BD más rápida";
    printf("El valor de A es: %d<br>", $A);
    printf("Ahora relleno con ceros: %03d<br>", $A);
    printf("El valor de B es: %f<br>", $B);
    printf("Ahora relleno con ceros: %01.2f<br>", $B);
    printf("El valor de C es: %s<br>", $C);
?>
</font>
```

El resultado es el siguiente:

El valor de A es: 10 Ahora relleno con ceros: 010 El valor de B es: 15.500000 Ahora relleno con ceros: 15.50 El valor de C es: MySQL la BD más rápida

Constantes

PHP define varias constantes y proporciona un mecanismo para definir más en tiempo de ejecución. Las constantes son como las variables, salvo por dos circunstancias, que las constantes deben ser definidas usando la función `define()` y no pueden ser redefinidas con otro valor.

Las constantes especiales `__FILE__` y `__LINE__` son una excepción, ya que actualmente no lo son. Las constantes son sensibles a mayúsculas, por convención, los identificadores de constantes suelen declararse en mayúsculas.

Las constantes sólo pueden contener valores escalares: boolean, integer, float y string.

Ejemplo 26

Archivo: `cap1\ejm26.php`

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    define("PI",3.141516);
    $radio = 5;
    $area = PI * $radio * $radio;
    echo("PI: " . PI . "<br>");
    echo("Radio: $radio<br>");
    echo("Area: $area");
?>
</font>
```

El resultado es el siguiente:

PI: 3.141516
Radio: 5
Area: 78.5379

PHP ofrece un gran número de constantes predefinidas a cualquier programa en ejecución. Muchas de estas constantes, sin embargo, son creadas por diferentes extensiones, y solo estarán presentes si dichas extensiones están disponibles, bien por carga dinámica o porque han sido compiladas. Entre estas constantes podemos mencionar las siguientes:

Constante	Descripción
__FILE__	El nombre del archivo (programa.php) que está siendo interpretado actualmente. Si se usa dentro de un archivo que ha sido incluido o requerido, entonces se da el nombre del archivo incluido, y no el nombre del archivo padre.
__LINE__	El número de línea dentro del archivo que está siendo interpretado. Si se usa dentro de un archivo incluido o requerido, entonces se da la posición dentro del archivo incluido.
PHP_VERSION	La cadena que representa la versión del analizador de PHP en uso.
PHP_OS	El nombre del sistema operativo en el cuál se ejecuta el analizador PHP.
TRUE	Valor verdadero.
FALSE	Valor falso.
E_ERROR	Denota un error distinto de un error de interpretación del cual no es posible recuperarse.
E_WARNING	Denota una condición donde PHP reconoce que hay algo erróneo, pero continuará de todas formas; pueden ser capturados por el propio programa de PHP.
E_PARSE	El intérprete encontró sintaxis inválida en el programa PHP. La recuperación no es posible.
E_NOTICE	Ocurrió algo que pudo ser o no un error. La ejecución continúa.

Ejemplo 27

Archivo: cap1\ejm27.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    function msgError($file, $line, $message) {
        echo("<b>ERROR</b><br>");
        echo("<b>Archivo:</b> $file<br>");
        echo("<b>Linea:</b> $line<br>");
        echo("<b>Mensaje:</b> $message");
    }

    msgError(__FILE__, __LINE__, "Algo está mal!!!");
?>
</font>
```

El resultado es el siguiente:

<p>ERROR Archivo: C:\wamp\www\prog2\sem01\ejm27.php Linea: 15 Mensaje: Algo está mal!!!</p>

Ejemplo 28

Archivo: cap1\ejm28.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
  <body text="Navy">
    <table border="2" width="400">
      <tr>
        <th>Constante</th>
        <th>Valor</th>
      </tr>
      <tr>
        <td>PHP_VERSION</td>
        <td><?php echo( PHP_VERSION ) ?></td>
      </tr>
      <tr>
        <td>PHP_OS</td>
        <td><?php echo( PHP_OS ) ?></td>
      </tr>
    </table>
  </body>
</font>
```

El resultado es el siguiente:

Constante	Valor
PHP_VERSION	5.5.12
PHP_OS	WINNT

EXPRESIONES Y OPERADORES

Sin duda alguna, las expresiones constituyen la base de todo lenguaje, estas se construyen en base a operadores.

Operadores Aritméticos

Recordemos la aritmética básica de colegio.

Ejemplo	Nombre	Resultado
$\$a + \b	Adición	Suma de $\$a$ y $\$b$.
$\$a - \b	Substracción	Diferencia entre $\$a$ y $\$b$.
$\$a * \b	Multiplicación	Producto de $\$a$ por $\$b$.
$\$a / \b	División	Cociente de $\$a$ entre $\$b$.
$\$a \% \b	Módulo	Resto de $\$a$ dividido entre $\$b$.

Operadores de Asignación

Estos operadores permiten asignar un valor a una variable.

Operador	Ejemplo	Equivalente
=	$\$a = 7;$	
+=	$\$a += 3;$	$\$a = \$a + 3;$
-=	$\$a -= 3;$	$\$a = \$a - 3;$
*=	$\$a *= 3;$	$\$a = \$a * 3;$
/=	$\$a /= 3;$	$\$a = \$a / 3;$
%=	$\$a \% = 3;$	$\$a = \$a \% 3;$

Ejemplo 29

Archivo: cap1\ejm29.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $a = 10;
    $b = 40;
    echo("\$a = $a<br>");
    echo("\$b = $b<br>");
    $b += $a;
    echo("El nuevo valor de \$b es $b");
?>
</font>
```

El resultado es el siguiente:

<pre>\$a = 10 \$b = 40 El nuevo valor de \$b es 50</pre>
--

Operadores de Comparación

Permiten comparar dos valores; el resultado es un valor de tipo **boolean**, TRUE o FALSE.

Ejemplo	Nombre	Resultado
<code>\$a == \$b</code>	Igualdad	Cierto si \$a es igual a \$b.
<code>\$a === \$b</code>	Identidad	Cierto si \$a es igual a \$b y si son del mismo tipo (sólo PHP4 ó superior)
<code>\$a != \$b</code>	Desigualdad	Cierto si \$a no es igual a \$b.
<code>\$a < \$b</code>	Menor que	Cierto si \$a es estrictamente menor que \$b.
<code>\$a > \$b</code>	Mayor que	Cierto si \$a es estrictamente mayor que \$b.
<code>\$a <= \$b</code>	Menor o igual que	Cierto si \$a es menor o igual que \$b.
<code>\$a >= \$b</code>	Mayor o igual que	Cierto si \$a es mayor o igual que \$b.

Otro operador condicional es el operador "?:" (o ternario), que funciona como en C y otros muchos lenguajes.

```
(expr1) ? (expr2) : (expr3);
```

La expresión toma el valor **expr2** si **expr1** se evalúa como verdadera, y toma el valor **expr3** si **expr1** se evalúa como falso.

Operador de Ejecución

PHP soporta un operador de ejecución: el apóstrofe invertido (`). ¡No son apóstrofes normales! PHP intentará ejecutar la instrucción contenida dentro de los apóstrofes invertidos como si fuera un comando del shell; y su salida devuelta como el valor de esta expresión.

Ejemplo 30

Archivo: cap1\ejm30.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $output = `dir ejm*`;
    echo("<pre>$output</pre>");
?>
</font>
```

Una parte del resultado se ilustra a continuación:

```
El volumen de la unidad C es SYSTEM
El número de serie del volumen es: DAE1-5300

Directorio de C:\wamp\www\prog2\sem02

21/07/2016  06:18 a.m.          411 ejm01.php
21/07/2016  06:18 a.m.          405 ejm02.php
21/07/2016  06:21 a.m.          422 ejm03.php
21/07/2016  06:23 a.m.          415 ejm04.php
14/08/2016  04:52 p.m.          220 ejm05.php
14/08/2016  04:55 p.m.          281 ejm06.php
21/07/2016  05:37 p.m.          370 ejm07.php
14/08/2016  10:09 p.m.          441 ejm08.php
21/07/2016  08:25 p.m.          415 ejm09.php
21/07/2016  05:14 p.m.          220 ejm10.php
21/07/2016  11:30 p.m.          324 ejm11.php
22/07/2016  01:03 a.m.          554 ejm12.php
22/07/2016  01:21 a.m.          316 ejm13.php
21/07/2016  11:56 p.m.          272 ejm14.php
14/08/2016  09:18 p.m.          454 ejm15.php
```

Operadores de Incremento/Decremento

PHP soporta los operadores de pre y post incremento y decremento, similar C.

Ejemplo	Nombre	Efecto
++\$a	Pre-incremento	Incrementa \$a en uno y después devuelve \$a.
\$a++	Post-incremento	Devuelve \$a y después incrementa \$a en uno.
--\$a	Pre-decremento	Decrementa \$a en uno y después devuelve \$a.
\$a--	Post-decremento	Devuelve \$a y después decrementa \$a en uno.

Ejemplo 31

Archivo: cap1\ejm31.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $a = 15;
    echo( "\$a = " . ++$a . "<br>" );
    echo( "\$a = " . $a++ . "<br>" );
    echo( "\$a = " . $a );
?>
</font>
```

El resultado es el siguiente:

\$a = 16
\$a = 16
\$a = 17

Operadores Lógicos

Permiten construir expresiones lógicas compuestas.

Ejemplo	Nombre	Resultado
$a \text{ and } b$	Y	Verdadero si a y b son verdaderos.
$a \text{ or } b$	O	Verdadero si a o b es verdadero.
$a \text{ xor } b$	O exclusiva	Verdadero si a es verdadero o b es verdadero, pero no ambos a la vez.
$\neg a$	Negación	Verdadero si a es falso.
$a \ \&\& \ b$	Y	Verdadero si a y b son verdaderos.
$a \ \ b$	O	Verdadero si a o b es verdadero

Ejemplo 32

Archivo: cap1\ejm32.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $a = rand(0,20);
    $b = rand(0,20);
    if($a>14 and $b>14){
        $cond = "Condición: Aprobado";
    }
    else{
        $cond = "Condición: Desaprobado";
    }
    echo( "\$a = $a<br>" );
    echo( "\$b = $b<br>" );
    echo( $cond );
?>
</font>
```

El resultado es el siguiente:

<p>\$a = 16 \$b = 19 Condición: Aprobado</p>
--

Operadores de Cadena

Operadores de Concatenación

Para concatenar dos cadenas utilizamos el operador Punto (.).

Ejemplo 33

Archivo: cap1\ejm33.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Helvetica">
<?php
    $cad1 = "PHP is ";
    $cad2 = "PowerFull.";
    echo( $cad1 . $cad2 );
?>
</font>
```

El resultado es el siguiente:

PHP is PowerFull.

Operador de Concatenación y Asignación

Este es el operador punto e igual (.=), agrega a una cadena, otra cadena.

Ejemplo 34

Archivo cap1\ejm34.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<font face="Arial">
<?php
    $cad = "Este es el equipo: ";
    $cad .= "Gustavo, Sergio, y Ricardo";
    echo( $cad );
?>
</font>
```

El resultado es el siguiente:

Este es el equipo: Gustavo, Sergio, y Ricardo

PRÁCTICA DE LABORATORIO

Revisar la instalación de PHP

Desarrollo el siguiente programa:

Programa: `info.php`

```
<?php
    Phpinfo();
?>
```

Luego de ejecutar el programa `info.php` proceda a revisar la configuración de PHP.

Generar Tabla de Multiplicar

Tomando como base el ejemplo 8, desarrolle un programa que genere la tabla de multiplicar del número **N**.

El valor de **N** debe ser generado aleatoriamente.

División de Dos Números

Desarrolle un programa que permita encontrar el cociente y el residuo de una división entera.

Capítulo 2

VARIABLES EXTERNAS

FORMULARIOS HTML (GET Y POST)

Los diferentes documentos que conforman una aplicación Web necesitan comunicarse entre sí, una de las formas es utilizando formularios HTML que envían datos a un programa PHP, estos campos enviados se convierten en variables dentro el programa PHP, y como provienen de otro documento se les denomina **Variables Externas**.

Esta forma sencilla que proporciona el lenguaje PHP de manejar formularios, nos permite procesar información que el usuario ingresa a través de nuestra aplicación Web.

Sintaxis

```
<FORM METHOD="POST/GET" ACTION="destino" ACCEPT-CHARSET="UTF-8">
```

Controles HTML

```
</FORM>
```

En este libro asumo que usted ya maneja HTML y Java Script, en todo caso recomiendo conseguir un manual y leer sobre estos lenguajes.

Ejemplo 35

Archivo: cap2\ejm35.html

```
<h1> Registro de Clientes</h1>
<form method="POST" action="ejm01.php" accept-charset="UTF-8">
  Nombre
  <input type="text" name="txtnombre" size="20" maxlength="20"><br>
  Email
  <input type="text" name="txtemail" size="20" maxlength="20"><br>
  <input type="submit" value="Enviar">
  <input type="reset" value="Limpiar">
</form>
```

El resultado es:

Registro de Clientes

Nombre

Email

Enviar

Limpiar

Los campos del formulario están automáticamente disponibles dentro del programa PHP, el nombre de estas variables toman el siguiente formato:

```
$NombreDelCampo
```

El uso de este método sólo es posible si la directiva **REGISTER_GLOBALS** está en ON.

Según el método que utilicemos (POST o GET) tenemos dos arreglos globales (\$_POST y \$_GET) que nos permiten recibir los campos que son enviados desde el formulario, el uso de éstos arreglos es más seguro porque no depende del estado de la directiva REGISTER_GLOBALS.

Si utilizamos el método GET debemos utilizar el siguiente formato:

```
$_GET["NombreDelCampo"]
```

Y si utilizamos el método POST es el siguiente formato:

```
$_POST["NombreDelCampo"]
```

Si el campo puede ser pasado indistintamente utilizando el método GET o POST, podemos utilizar el arreglo global \$_REQUEST, éste arreglo no depende de la directiva REGISTER_GLOBALS y tampoco del método de envío, el formato para obtener el valor de los campos es el siguiente:

```
$_REQUEST["NombreDelCampo"]
```

DIRECTIVA REGISTER_GLOBALS

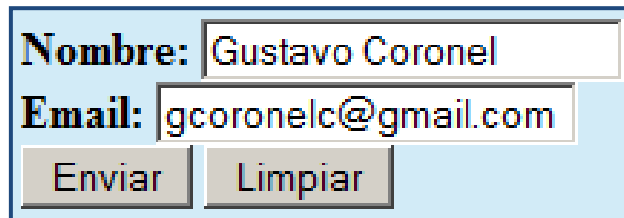
Al diseñar un formulario debemos indicar la página PHP que procesará el formulario, así como el método por el que se le pasará la información a la página, tal como se aprecia en el Ejemplo 2.

Ejemplo 36

Archivo: cap2\ejm36.html

```
<body bgcolor="#D2EBF7">
  <form method=post action="ejm03.php" accept-charset="UTF-8">
    <b>Nombre: </b>
    <input type="text" name="nombre" size="20" maxlength="20" ><br>
    <b>Email: </b>
    <input type="text" name="email" size="20" maxsize="20"><br>
    <input type="submit" value="Enviar">
    <input type="reset" value="Limpiar">
  </form>
</body>
```

El resultado es el siguiente:



Cuando se envía un formulario HTML a un programa PHP, los campos de dicho formulario pasan a estar automáticamente disponibles en el programa como variables.

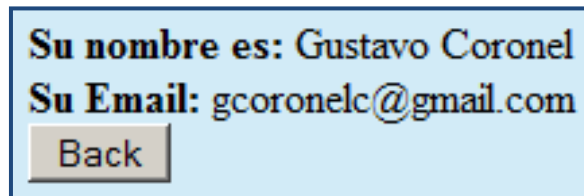
Es necesario revisar la directiva REGISTER_GLOBALS en el archivo **php.ini**, esta directiva debe estar habilitada (ON) para que PHP proceda a crear las variables externas de manera automática.

Ejemplo 37

Archivo: cap2\ejm37.php

```
<?php header('Content-Type: text/html; charset=utf-8' ); ?>
<body bgcolor="#D2EBF7">
  <?php
    echo("<b>Su nombre es:</b> $nombre<br>");
    echo("<b>Su Email:</b> $email<br>");
  ?>
  <input type="button" value="Back" onClick="history.back()">
</body>
```

El resultado es el siguiente:



ARREGLOS GLOBALES HTTP

Arreglo \$_POST

Se trata de un arreglo asociativo de variables pasadas al programa actual a través del método HTTP POST.

Esta es una variable **superglobal**, esto simplemente quiere decir que está disponible en todos los contextos a lo largo del programa.

El Ejemplo 4 (ejm04.html) muestra un formulario que permite el ingreso de datos para calcular el sueldo de un trabajador, y el Ejemplo 5 (ejm05.php) recibe los datos y realiza el cálculo del sueldo.

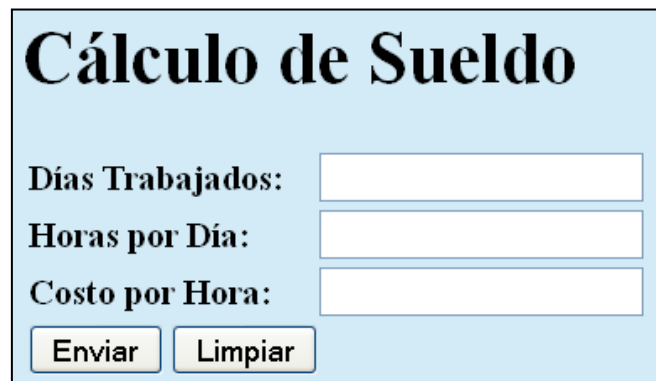
Ejemplo 38

Archivo: cap2\ejm38.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body bgcolor="#D2EBF7">
    <h1>Cálculo de Sueldo</h1>
    <form method="post" action="ejm05.php" accept-charset="UTF-8">
      <table width="317">
        <tr>
          <td width="125"><b>Días Trabajados:</b></td>
          <td width="180"><input type="text" name="dt"></td>
        </tr>
        <tr>
          <td><b>Horas por Día:</b></td>
          <td><input type="text" name="hd"></td>
        </tr>
        <tr>
          <td><b>Costo por Hora:</b></td>
          <td><input type="text" name="ch"></td>
        </tr>
        <tr>
          <td colspan="2">
            <input type="submit" value="Enviar">
            <input type="reset" value="Limpiar">
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

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

El resultado es el siguiente:



Cálculo de Sueldo

Días Trabajados:

Horas por Día:

Costo por Hora:

Ejemplo 39

Archivo: cap2\ejm39.php

```
<?php
// Datos
$dt = $_POST["dt"]; // Días trabajados
$hd = $_POST["hd"]; // Horas por día
$ch = $_POST["ch"]; // Costo po hora
// Proceso
$total = $dt * $hd * $ch;
$ir = $total * 0.10; # Impuesto a la renta
$neto = $total - $ir;
// Formatos
$ch = "S/. " . number_format($ch, 2, ".", "");
$total = "S/. " . number_format($total, 2, ".", "");
$ir = "S/. " . number_format($ir, 2, ".", "");
$neto = "S/. " . number_format($neto, 2, ".", "");
?>

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<script>
function boton_back(){
document.location.href = "ejm04.html";
```

```
    }  
    </script>  
</head>  
<body bgcolor="#D2EBF7">  
    <h1>Cálculo de Sueldo</h1>  
    <table border=1 width=241>  
        <tr>  
            <th align=center colspan=2 border=0>Datos</th>  
        </tr>  
        <tr>  
            <td width=53%><b>Días trabajados</b></td>  
            <td width="47%"><?php echo($dt) ?></td>  
        </tr>  
        <tr>  
            <td><b>Horas por día</b></td>  
            <td><?php echo($hd) ?></td>  
        </tr>  
        <tr>  
            <td><b>Costo por hora</b></td>  
            <td><?php echo($ch) ?></td>  
        </tr>  
        <tr>  
            <th align=center colspan=2 border=0>  
                Resultado  
            </th>  
        </tr>  
        <tr>  
            <td><b>Total</b></td>  
            <td><?php echo($total) ?></td>  
        </tr>  
        <tr>  
            <td><b>Impuesto Renta</b></td>  
            <td><?php echo($ir) ?></td>  
        </tr>  
        <tr>  
            <td><b>Neto</b></td>  
            <td><?php echo($neto) ?></td>  
        </tr>  
    </table>  
    <input type="button" value="Retornar" onClick="boton_back()>  
</body>  
</html>
```

El resultado se muestra a continuación:

Cálculo de Sueldo	
Datos	
Días trabajados	6
Horas por día	8
Costo por hora	S/. 35.00
Resultado	
Total	S/. 1,680.00
Impuesto Renta	S/. 168.00
Neto	S/. 1,512.00
<input type="button" value="Retornar"/>	

Arreglo \$_GET

Se trata de un arreglo asociativo de variables pasadas al programa actual a través del método HTTP GET.

Esta es una variable **superglobal**, esto simplemente quiere decir que está disponible en todos los contextos a lo largo del programa.

El Ejemplo 6 (ejm06.php) muestra un formulario que permite el ingreso de dos números, estos datos son enviados al programa del Ejemplo 7 (ejm07.php), este programa recibe los datos, calcula la suma y muestra los resultados.

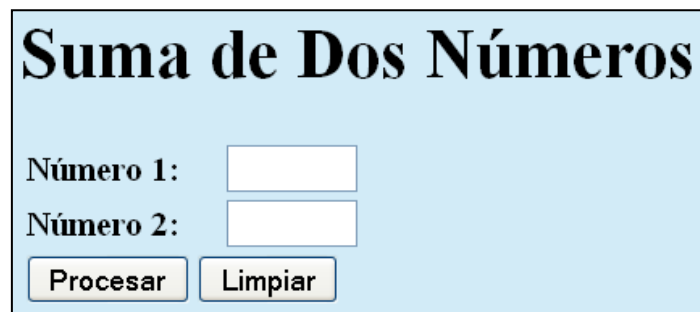
Ejemplo 40

Archivo: cap2\ejm40.html

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
  <h1>Suma de Dos Números</h1>
  <form method="get" action="ejm07.php">
    <table width="165">
      <tr>
```

```
<td width="88"><b>Número 1:</b></td>
<td width="65">
  <input name="N1" type="text" size="6" maxlength="6">
</td>
</tr>
<tr>
  <td><b>Número 2:</b></td>
  <td><input name="N2" type="text" size="6" maxlength="6"></td>
</tr>
<tr>
  <td colspan="2"><input type="submit" value="Procesar">
    <input type="reset" value="Limpiar"></td>
</tr>
</table>
</form>
</body>
</html>
```

El resultado es:



Suma de Dos Números

Número 1:

Número 2:

Ejemplo 41

Archivo: cap2\ejm41.php

```
<?php
// Datos
$n1 = $_GET["N1"];
$n2 = $_GET["N2"];
// Proceso
$suma = $n1 + $n2;
?>
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
```



```
<body bgcolor="#D2EBF7">
  <h1>Resultado</h1>
  <table width="168">
    <tr>
      <td width="66"><b>N1</b></td>
      <td width="135"><?php echo($n1) ?></td>
    </tr>
    <tr>
      <td><b>N2</b></td>
      <td><?php echo($n2) ?></td>
    </tr>
    <tr>
      <td><b>Suma</b></td>
      <td><?php echo($suma) ?></td>
    </tr>
  </table>
  <a href="ejm06.html">Nueva Suma</a>
</body>
</html>
```

El resultado es:

Resultado	
N1	35
N2	76
Suma	111
Nueva Suma	

El método GET también se utiliza con QUERY_STRING, como ejemplo ilustrativo anote la siguiente URL en el campo **Dirección** del su navegador:

```
http://localhost/cap2/ejm41.php?N1=78&N2=65
```

Como podemos observar en la misma URL podemos especificar los campos.

PROGRAMAS RECURSIVOS

En los ejemplos anteriores hemos visto que el formulario está en un documento (*.html) y el proceso en otro documento (*.php), también tenemos la opción de hacerlo en un solo documento, donde el formulario envía los datos al mismo documento, el esquema es el siguiente:

```
<?php if(!isset($_POST["control"])){  ?>

    <form method="POST" action="destino.php">

        ----
        ----
        ----

    </form>

<?php } else {

    // Proceso de Datos

} ?>
```

En este caso **control** sería un campo más del formulario, pero oculto para que no se muestre al usuario, y permite verificar si se debe mostrar el formulario ó procesar los datos. También se puede realizar utilizando el método GET, aunque por seguridad se recomienda utilizar el método POST, y para recoger los datos se puede utilizar el arreglo \$_REQUEST independiente del método de envío (GET ó POST).

Ejemplo 42

Este ejemplo permite calcular el área de un triángulo.

Archivo: cap2\ejm42.php

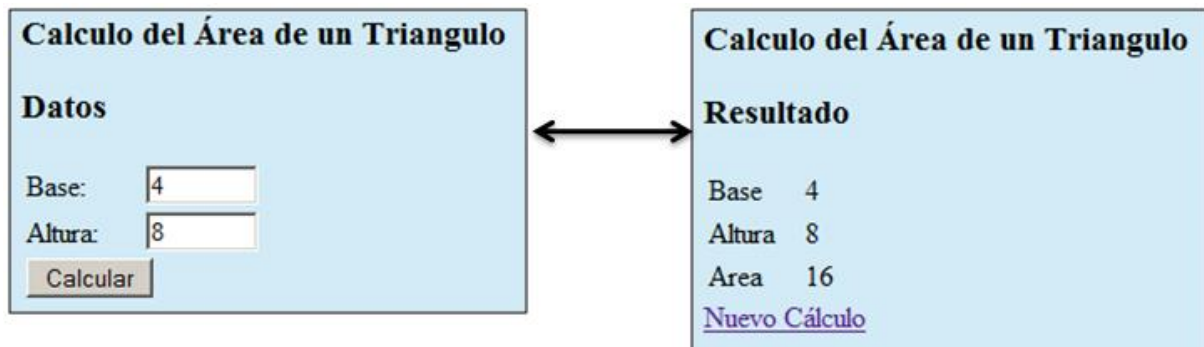
```
<?php
$formulario = TRUE;
if( isset( $_REQUEST["control"] ) ){
    // Datos
    $base = $_REQUEST["base"];
    $altura = $_REQUEST["altura"];
```

```
// Proceso
$area = $base * $altura / 2;
$formulario = FALSE;
}
?>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
  </head>
  <body bgcolor="#D2EBF7">
    <h1>Calculo del Área de un Triangulo</h1>
    <?php if( $formulario ) { ?>
      <h3>Datos</h3>
      <form method="post" action="ejm08.php">
        <input type="hidden" name="control" value="12345">
        <table width="164">
          <tr>
            <td width="62">Base:</td>
            <td width="90">
              <input name="base" type="text" size="6" maxlength="6">
            </td>
          </tr>
          <tr>
            <td>Altura:</td>
            <td><input name="altura" type="text" size="6" maxlength="6"></td>
          </tr>
          <tr>
            <td colspan="2"><input type="submit" value="Calcular"></td>
          </tr>
        </table>
      </form>
      <?php } else { ?>
        <h3>Resultado</h3>
        <table width="138">
          <tr>
            <td width="49">Base</td>
            <td width="77"><?php echo($base) ?></td>
          </tr>
          <tr>
            <td>Altura</td>
            <td><?php echo($altura) ?></td>
          </tr>
          <tr>
            <td>Area</td>
            <td><?php echo($area) ?></td>
          </tr>
        </table>
      </form>
    </body>
  </html>
```

```
</tr>
</table>
<a href="ejm08.php">Nuevo Cálculo</a>
<?php } ?>
</body>
</html>
```

El resultado se ilustra a continuación:



COOKIES HTTP

PHP soporta cookies HTTP de forma transparente tal y como están definidas en las Netscape's Spec. Las cookies son un mecanismo para almacenar datos en el navegador y así rastrear o identificar a usuarios que vuelven a ingresar a nuestro sitio Web.

Se pueden crear cookies usando la función `SetCookie()`. Las cookies son parte de la cabecera HTTP, así que se debe llamar a la función `SetCookie` antes de que se envíe cualquier salida al navegador. Los datos de una cookie están disponibles en el arreglo con datos de cookies apropiada, tal como `$_COOKIE`.

Ejemplo 43

Archivo: cap2\ejm43.php

```
<?php
$primera = 0;
if( !isset($_COOKIE["nombre"]) ) {
    setcookie("nombre","Claudia",time()+3600);
    $primera = 1;
}
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body bgcolor="#D2EBF7">
    <?php
    if($primera){
        echo "Hola, esta es tu primera visita.";
    }else{
        echo "Hola " . $_COOKIE["nombre"] . "<br>";
    }
    ?>
</body>
</html>
```

La primera vez que ejecute el programa el resultado es el siguiente:

Hola, esta es tu primera visita.

A partir de la segunda vez, el resultado será este:

Hola Claudia

Para que las cookies funcionen el navegador debe soportarlas y además tienen que estar habilitadas.

PROYECTOS PROPUESTOS

Proyecto 1

La empresa "EVENTOS PERU" está realizando una serie de presentaciones con artistas nacionales y extranjeros en varios locales de todo el Perú.

La empresa necesita de una aplicación para calcular la rentabilidad del evento, para lo cual se ha considerado lo siguiente:

1. El evento tiene una categoría, que puede ser: LOCAL, INTERNACIONAL o PREMIUM.
2. El local donde se realiza el evento debe tener una capacidad mínima de 15,000 personas, puede ser más.
3. El precio de la entrada al evento y el pago al artista está en función a la categoría del evento según el siguiente cuadro:

CATEGORIA	PRECIO DE ENTRADA	PAGO AL ARTISTA
LOCAL	S/. 10.00	S/. 30,000.00
INTERNACIONAL	S/. 20.00	S/. 70,000.00
PREMIUM	S/. 30.00	S/. 100,000.00

Además, se debe considerar los siguientes gastos:

1. Publicidad: 7% del ingreso bruto
2. Logística: 8% del ingreso bruto
3. Transporte: 4% del ingreso bruto (Solo si el evento es en provincia)

Desarrollar la aplicación que requiere la empresa "EVENTOS PERU" para estimar la rentabilidad de un evento. La aplicación debe mostrar un reporte de los INGRESOS, GASTOS y la RENTABILIDAD con sus respectivos detalles.

Capítulo 3

ESTRUCTURAS DE CONTROL

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>

ESTRUCTURAS CONDICIONALES

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 44

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

Archivo: cap3\ejm44.html

```
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
  </head>  
  <body bgcolor="#D2EBF7">  
    <form method = "post" action = "ejm45.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>
```

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

El resultado:

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

Ejemplo 45

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.

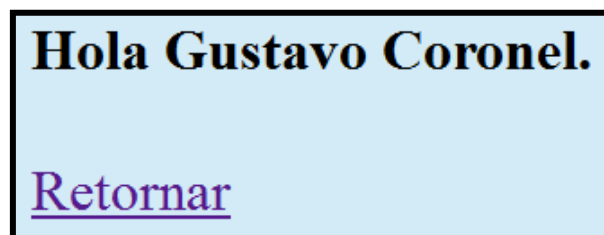
Archivo: cap3\ejm45.php

```
<?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:



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 46

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: cap3\ejm46.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 ) {
```

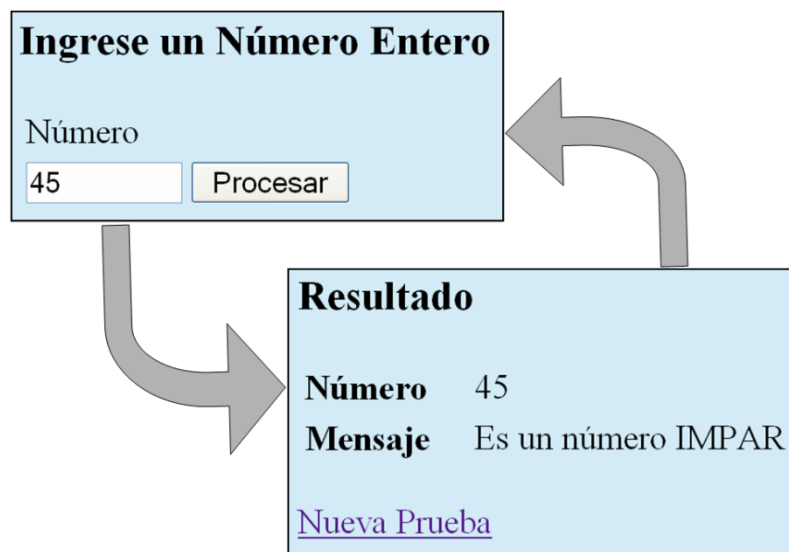
```
        $msg = "El número ingresado no es un entero";
    } else {
        $n = (int)$n;
        $r = $n % 2;
        if($r==0){
            $msg = "Es un número PAR";
        } else {
            $msg = "Es un número IMPAR";
        }
    }
}
?>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body bgcolor="#D2EBF7">
        <?php if( $mostrar == "formulario" ) { ?>
            <div id="formulario">
                <h3>Ingresa un Número Entero</h3>
                <form method = "post" action = "ejm03.php" accept-charset="utf-8">
                    <input type="hidden" name="seguro" value="alianza">
                    <table width="172">
                        <tr>
                            <td colspan="2">Número</td>
                        </tr>
                        <tr>
                            <td width="66">
                                <input type="text" name="num" size="8" maxlength="8">
                            </td>
                            <td width="94"><input type="submit" value="Procesar"></td>
                        </tr>
                    </table>
                </form>
            </div>
            <?php } ?>
            <?php if( $mostrar == "resultado" ) { ?>
                <div id="resultado">
                    <h3>Resultado</h3>
                    <table width="500">
                        <tr>
                            <td width="74"><b>Número</b></td>
                            <td width="414"><?php echo($n) ?></td>
                        </tr>
                    </table>
                </div>
            </div>
        </div>
    </body>
</html>
```

```
<td><b>Mensaje</b></td>
<td><?php echo($msg) ?></td>
</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:



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 47

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: cap3\ejm47.php

```
<?php
$mostrar = "formulario";
if( isset( $_POST["seguro"] ) ) {
    $mostrar = "resultado"; // Existe el dato
}
if( $mostrar == "resultado" ) {
    // Datos
    $n1 = (float)$_POST["n1"];
    $n2 = (float)$_POST["n2"];
    // Proceso
    $pr = ($n1 + $n2) / 2;
    if ($pr >= 18) {
        $cond = "Excelente";
    } elseif ($pr >= 15) {
        $cond = "Bueno";
    } elseif ($pr >= 11) {
        $cond = "Regular";
    } elseif ($pr >= 6) {
        $cond = "Malo";
    } else {
        $cond = "Pésimo";
    }
    // formato
    $n1 = number_format($n1, 2);
    $n2 = number_format($n2, 2);
    $pr = number_format($pr, 2);
}
?>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    </head>
    <body bgcolor="#D2EBF7">
        <h3>Cálculo de Promedio</h3>
        <?php if( $mostrar == "formulario" ) { ?>
            <div id="formulario">
                <form method = "post" action = "ejm04.php" accept-charset="utf-8">
                    <input type="hidden" name="seguro" value="alianza">
                    <strong>Ingreso de Notas</strong>
                    <table width="218">
                        <tr>
                            <td width="56">Nota 1</td>
```

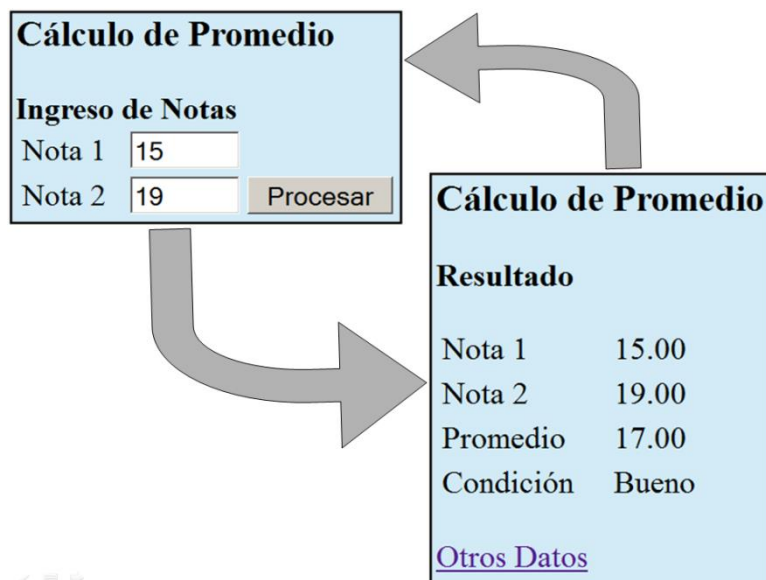


```
<td width="45">
    <input type="text" name="n1" size="5" maxlength="2">
</td>
<td width="101">&nbsp;  </td>
</tr>
<tr>
<td>Nota 2</td>
<td><input type="text" name="n2" size="5" maxlength="2"></td>
<td><input type="submit" value="Procesar"></td>
</tr>
</table>
</form>
</div>
<?php } ?>
<?php if( $mostrar == "resultado" ) { ?>
<div id="resultado">
    <p><strong>Resultado</strong></p>
    <table width="281">
        <tr>
            <td width="81">Nota 1</td>
            <td width="188"><?php echo($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:



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 48

Archivo: cap3\ejm48.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 49

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: cap3\ejm49.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 50

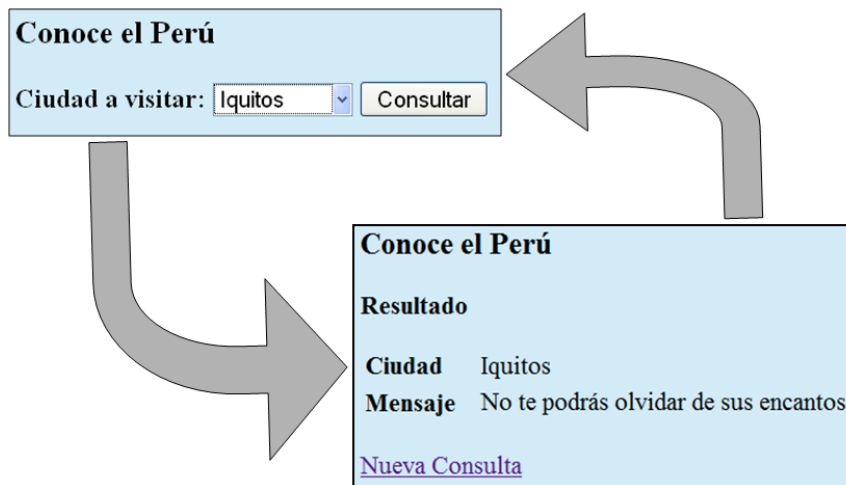
El siguiente ejemplo muestra un formulario para seleccionar un destino turístico, luego muestra un mensaje sobre el destino seleccionado.

Archivo: cap3\ejm50.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.

ESTRUCTURAS REPETITIVAS

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 51

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: cap3\ejm51.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: 14
Número 2: 19

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

[Nueva Prueba](#)

Ejemplo 52

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.

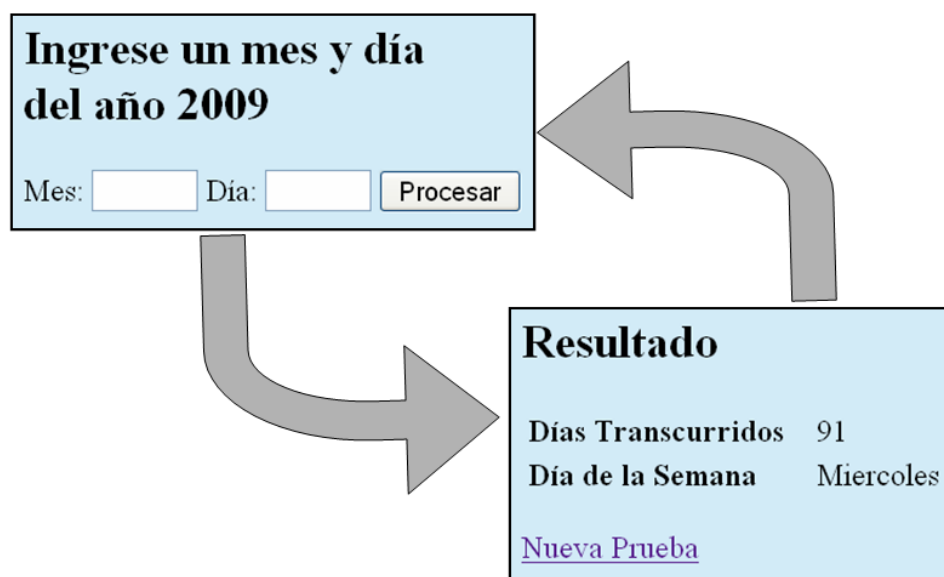
Archivo: cap3\ejm52.php

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

```
</tr>
</table>
<p><a href='ejm09.php'>Nueva Prueba</a></p>
</div>
<?php } ?>
</body>
</html>
```

Su ejecución se ilustra a continuación:



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 53

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

Archivo: `cap3\ejm53.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:

The diagram illustrates the process of generating a multiplication table based on user input. It consists of two main components: a user input form and a multiplication table.

User Input Form: A box titled "Tabla de Multiplicar" containing a text input field labeled "Número" with the value "5" and a button labeled "Procesar".

Multiplication Table: A box titled "Tabla de Multiplicar" containing a sub-header "Tabla del 5" and a table of multiplication results for the number 5. The table has 12 rows, each representing a multiplication of 5 by a number from 1 to 12. The results are as follows:

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

Below the table is a link labeled "Nueva Prueba".

Arrows indicate the flow of data: a straight arrow points from the "Procesar" button to the multiplication table, and a curved arrow points from the bottom of the multiplication table back to the "Número" input field, suggesting a feedback loop or a return to the input stage.

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
expresion1	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 for.
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**

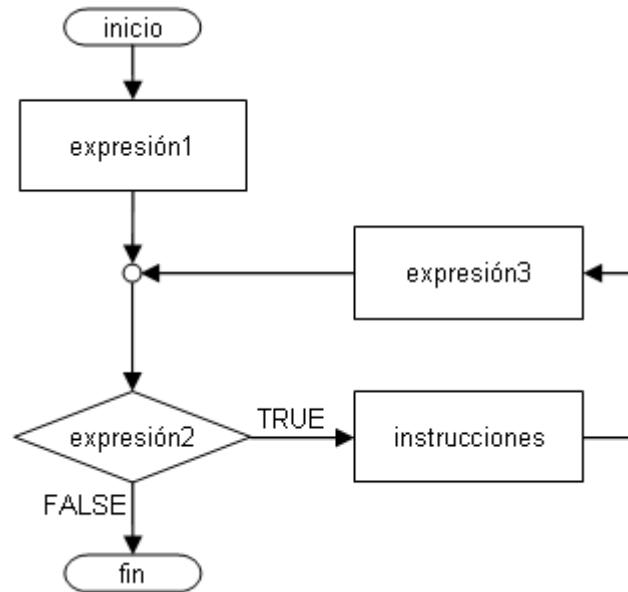


Figura 1 Diagrama de Flujo de la estructura **for**.

Ejemplo 54

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

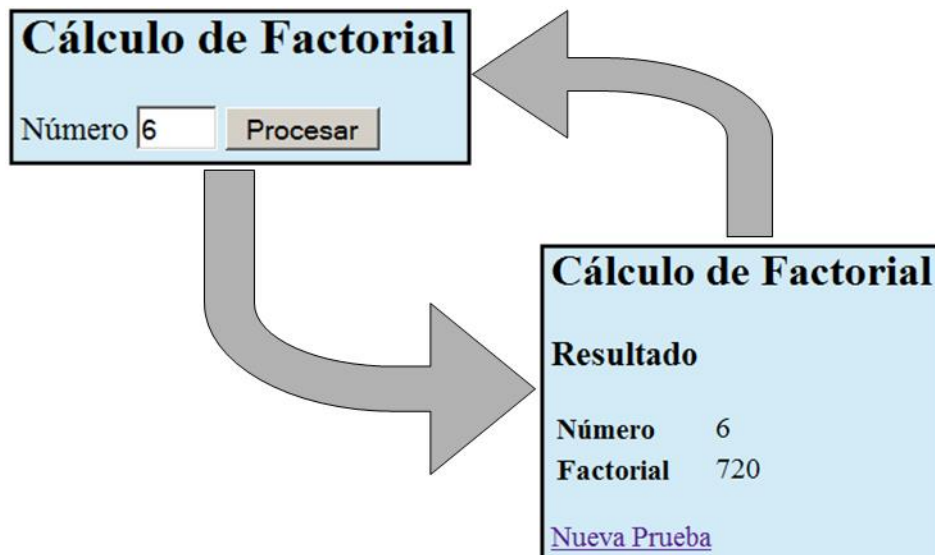
Archivo: cap3\ejm54.php

```
<?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:



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 55

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

Archivo: `cap3\ejm55.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 56

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

Archivo: cap3\ejm56.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

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 57

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

Archivo: cap3\ejm57.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 58

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

Archivo: `cap3\ejm58.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 59

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

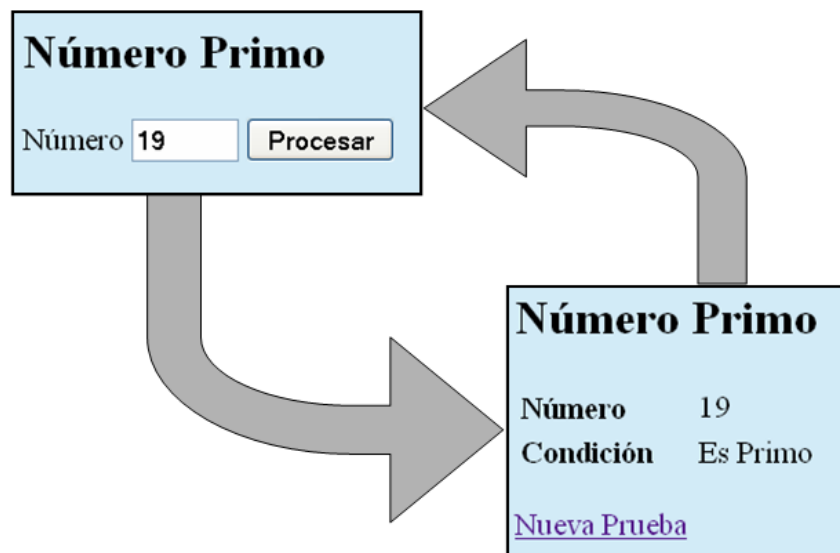
Archivo: cap3\ejm59.php

```
<?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 60

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: cap3\ejm60.php

```
<?php

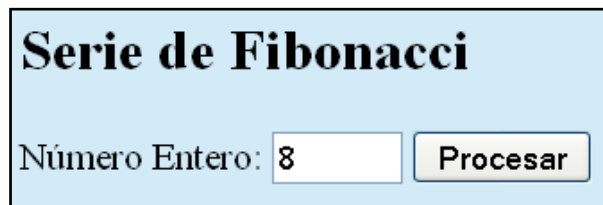
$mostrar = FALSE;
if( isset($_POST["procesar"]) ) {
    $mostrar = TRUE;
    $num = $_POST["num"];
    if ($num <= 2){
        $msg = "Debe ingresar un número entero mayor que 2.";
    } else {
        $msg = "1 1";
        $a = 1;
        $b = 1;
        for( $k = 3; $k <= $num; $k++ ) {
          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

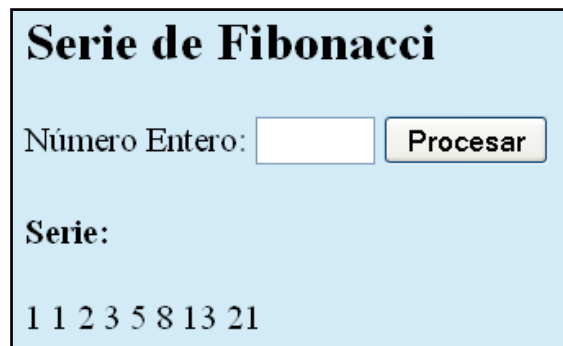
```
</div>  
<?php } ?>  
</body>  
</html>
```

Cuando ejecuta este programa inicialmente se muestra la siguiente interfaz:



The screenshot shows a web form titled "Serie de Fibonacci". It contains a label "Número Entero:" followed by a text input field containing the number "8". To the right of the input field is a button labeled "Procesar".

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



The screenshot shows the same web form after processing. The "Número Entero:" input field is now empty. Below the input field, the text "Serie:" is displayed, followed by the sequence of numbers "1 1 2 3 5 8 13 21". The "Procesar" button remains visible.

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 61

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: cap3\ejm61.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:



Movimiento Bancario

Saldo Actual: 5000

Tipo: D - Depósito ▼

Importe: 100

Procesar limpiar

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 62

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

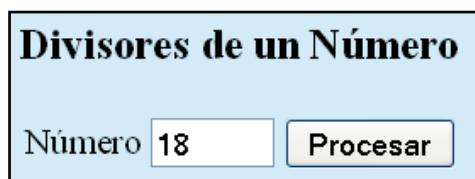
Archivo: cap3\ejm62.php

```
<?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:



Divisores de un Número

Número

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

Divisores de un Número

Número

Resultado

Número: 18
Divisores: 2 3 6 9

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

Ejemplo 63

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.

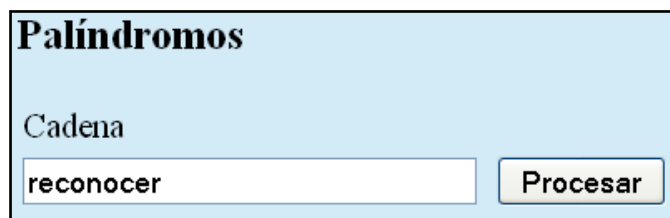
Archivo: cap3\ejm63.php

```
<?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 este programa:



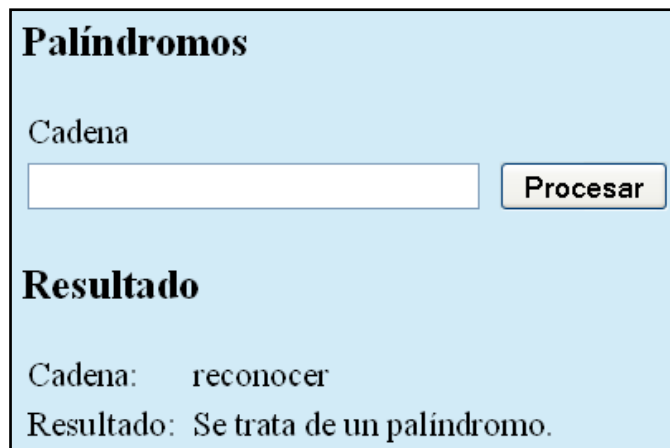
Palíndromos

Cadena

reconocer

Procesar

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



Palíndromos

Cadena

Procesar

Resultado

Cadena: reconocer

Resultado: Se trata de un palíndromo.

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

PROYECTOS PROPUESTOS

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.

Proyecto 2

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

Proyecto 3

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

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.

Capítulo 4

MANEJO DE LIBRERIAS

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 64

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

Archivo: cap4\ejm64.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 65

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

Archivo: cap4\ejm65.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>" );
    }
}

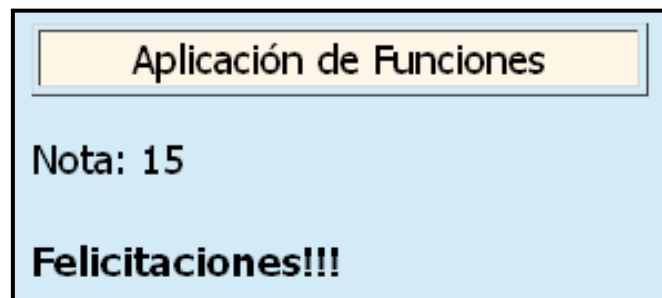
// *****
// Funciones Globales
// *****

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 66

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: cap4\ejm66.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

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:

4. Parámetros por valor
5. Parámetros por referencia
6. Parámetros por defecto
7. Número variable de parámetros

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 67

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: cap4\ejm67.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	

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 68

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: cap4\ejm68.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>
<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	

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 69

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: cap4\ejm69.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>
    </table>
</body>
</html>
```

```
<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	

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 70

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: cap4\ejm70.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	

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 71

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: cap4\ejm71.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 72

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: cap4\ejm72.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("$value ");  
    }  
    ?>  
</td>  
</tr>  
</table>  
<a href="<?php echo($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	

INCLUIR ARCHIVOS

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.

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 73

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: `cap4\titulo.html`

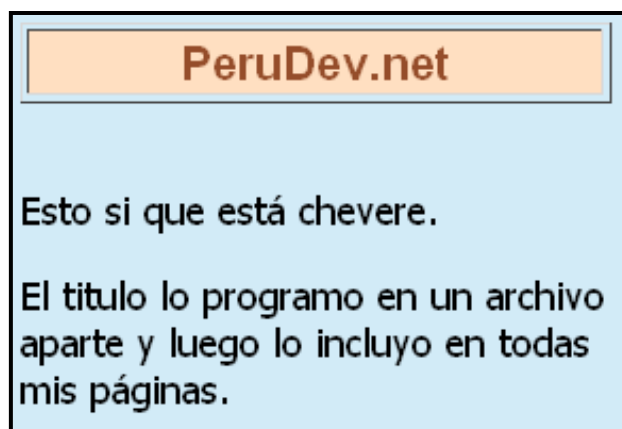
```
<style>
.titulo{
    background-color: #FFDFC1;
    font-family: Arial, Helvetica, sans-serif;
    font-weight: bold;
    font-size: 20px;
    text-align: center;
    vertical-align: middle;
    color: #954D2A;
}
</style>
<table border='1' width='250'>
```

```
<tr>
  <td class="titulo">
    PeruDev.net
  </td>
</tr>
</table>
<br>
```

Archivo: cap4\ejm73.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 **ejm73.php** en el navegador el resultado que obtenemos es:



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 74

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.
ejm74.php	Programa donde se incluirán los archivos anteriores.

Archivo: cap4\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: cap4\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: cap4\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: cap4\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: cap4\ejm74.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($a) ?></td>
  </tr>
  <tr>
    <td>Lado B</td>
    <td><?php echo($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 ejm74.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

Ingrese 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.

Capítulo 5

ARREGLOS

¿QUÉ ES UN ARREGLO?

Arreglo (Array en inglés) es una estructura de datos conformada por un conjunto de variables, agrupadas bajo un mismo nombre, a las cuales accedemos mediante un índice.

\$lista	
0	"Impresora"
1	100
2	550.00

Figura 2 Ejemplo de un arreglo.

El primer elemento del arreglo inicia con índice 0, y los elementos que conforman el arreglo pueden ser de cualquier tipo.

Ejemplo 75

Este ejemplo crea un arreglo, luego muestra cada uno de sus elementos y su respectivo tipo de dato.

Archivo: cap5\ejm75.php

```
<?php
// Creación del arreglo
$lista[] = "Impresora";
$lista[] = 100;
$lista[] = 165.79;
?>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Capítulo 06 - Ejemplo 01</title>
  </head>
```

```
<body bgcolor="#D2EBF7">
  <h3>Arreglos</h3>
  <?php
    echo( "\$lista[0] = " . $lista[0] . "<br>" );
    echo( "\$lista[1] = " . $lista[1] . "<br>" );
    echo( "\$lista[2] = " . $lista[2] . "<br>" );

    echo( "<br>" );
    echo( "\$lista[0] es de tipo " . gettype($lista[0]) . "<br>" );
    echo( "\$lista[1] es de tipo " . gettype($lista[1]) . "<br>" );
    echo( "\$lista[2] es de tipo " . gettype($lista[2]) . "<br>" );
  ?>
</body>
</html>
```

Resultado:

Arreglos

\$lista[0] = Impresora

\$lista[1] = 100

\$lista[2] = 165.79

\$lista[0] es de tipo string

\$lista[1] es de tipo integer

\$lista[2] es de tipo double

Como podemos apreciar, para acceder a un elemento utilizamos un índice que inicia en cero (0), en este caso el primer elemento corresponde a una variable de tipo *string*, el segundo elemento a una variable de tipo *integer*, y finalmente el tercer elemento a una variable de tipo *double*.

ARREGLOS UNIDIMENSIONALES

Un arreglo unidimensional es una lista de valores a los cuales accedemos mediante un índice que inicia en cero.

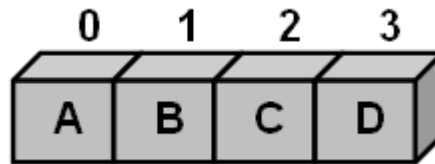


Figura 3 Representación gráfica de un arreglo unidimensional.

Para crear un arreglo tenemos varias alternativas, por ejemplo para iniciar la creación de una lista de frutas, la instrucción sería:

```
$lista[] = "Pera";
```

En este caso el índice de este primer elemento es cero (0), para agregar otro elemento utilizamos una instrucción similar:

```
$lista[] = "Naranja";
```

Para saber el tamaño de un arreglo podemos utilizar la función `count()`, tal como se ilustra a continuación:

```
$n = count($lista);
```

Ejemplo 76

El siguiente programa ilustra el uso de la función `count()` y cómo podemos hacer un recorrido por los elementos de un arreglo.

Archivo: cap5\ejm76.php

```
<?php
// Creación del Arreglo
$lista[] = "Pera";
$lista[] = "Naranja";
```



```
$lista[] = "Uva";  
$lista[] = "Manzana";  
?>  
  
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
    <title>Capítulo 06 - Ejemplo 02</title>  
  </head>  
  <body bgcolor="#D2EBF7">  
    <h3>Arreglos</h3>  
    <?php  
      $n = count($lista);  
      for( $j=0; $j < $n; $j++ ) {  
        echo( "\$lista[$j] = $lista[$j] <br>" );  
      }  
    ?>  
  </body>  
</html>
```

El resultado:

Arreglos

```
$lista[0] = Pera  
$lista[1] = Naranja  
$lista[2] = Uva  
$lista[3] = Manzana
```

También podemos utilizar la función `array()` para crear un arreglo, tal como se ilustra a continuación:

```
$lista = array("Chiclayo","Trujillo","Piura","Tumbes");
```

Ejemplo 77

Este ejemplo crea un arreglo utilizando la función `array()`, luego lista todos sus elementos.

Archivo: Cap5\ejm77.php

```
<?php
// Creación del Arreglo
$lista = array("Chiclayo","Trujillo","Piura","Tumbes");
?>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Capítulo 06 - Ejemplo 03</title>
  </head>
  <body bgcolor="#D2EBF7">
    <h3>Conoce el Norte de Perú</h3>
    <?php
      $n = count($lista);
      for( $j=0; $j < $n; $j++ ) {
        echo "\$lista[$j] = $lista[$j] <br>";
      }
    ?>
  </body>
</html>
```

El resultado:

Conoce el Norte de Perú

```
$lista[0] = Chiclayo
$lista[1] = Trujillo
$lista[2] = Piura
$lista[3] = Tumbes
```

Para recorrer los elementos de un arreglo podemos utilizar un bucle haciendo referencia a su índice, pero necesitamos conocer el tamaño del arreglo. La estructura de control `foreach` nos facilita el recorrido por los elementos de un arreglo por qué no necesitamos conocer su tamaño, a continuación, se ilustra su sintaxis:

```
foreach( $arreglo as $elemento) {  
  
    instrucciones;  
  
}
```

Ejemplo 78

En el siguiente ejemplo se ilustra el uso de la estructura `foreach`.

Archivo: `cap5\ejm78.php`

```
<?php  
$lista = array("Arroz con Pato","Ceviche","Seco de Cabrito",  
    "Seco de Chavelo","Sudado de Mero", "Espesado");  
?>  
  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
        <title>Capítulo 06 - Ejemplo 04</title>  
    </head>  
    <body bgcolor="#D2EBF7">  
        <h3>Gastronomía Del Norte de Perú</h3>  
        <ol>  
            <?php  
            foreach( $lista as $dato ) {  
                echo( "<li>$dato</li>" );  
            }  
            ?>  
        </ol>  
    </body>  
</html>
```

Resultado:

Gastronomía Del Norte de Perú

1. Arroz con Pato
2. Ceviche
3. Seco de Cabrito
4. Seco de Chavelo
5. Sudado de Mero
6. Espesado

ARREGLOS MULTIDIMENSIONALES

Si cada elemento de un arreglo unidimensional es otro arreglo, entonces tenemos un arreglo de dos dimensiones, si los elementos de este segundo arreglo son otros arreglos, entonces tenemos un arreglo de tres dimensiones, y así sucesivamente.

Cuando se trata de dos dimensiones cada elemento en particular está identificado por dos índices, si fuese de tres dimensiones serían tres índices y así sucesivamente.

Veamos la siguiente asignación:

```
$lista = array(  
    array(45,67,80,45),  
    array(67,89,23,67),  
    array(12,87,72,54)  
);
```

Estamos creando un arreglo de dos dimensiones, de 3 filas por 4 columnas, para acceder a un elemento se debe usar la siguiente sintaxis:

```
$lista[filas][columna]
```

	0	1	2	3
0	45	67	80	45
1	67	89	23	67
2	12	87	72	54

Figura 4 Representación gráfica de un arreglo de dos dimensiones.

Si necesitamos hacer un recorrido por todo el arreglo deberíamos utilizar la siguiente plantilla:

```
for( $f = 0; $f < count( $lista); $f++ ) {  
  
    for( $c = 0; $c < count( $lista[ $f ] ); $c++ ) {
```

```
// Proceso a desarrollar, el acceso al elemento es así: $lista[$f][$c]

    }

}
```

Ejemplo 79

En este ejemplo se ilustra el recorrido de un arreglo de dos dimensiones, conocido también como matriz.

Archivo: cap5\ejm79.php

```
<?php
// Creando el Arreglo
$lista = array(
    array("Perú","Lima"),
    array("Bolivia","La Paz"),
    array("Brazil","Brasilia"),
    array("Argentina","Buenos Aires"),
    array("Colombia","Bogotá")
);
?>

<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>Capítulo 06 - Ejemplo 05</title>
</head>
<body bgcolor="#D2EBF7">
    <h3>Conoce América Latina</h3>
    <table width="300" border="1">
        <tr>
            <th width="128">País</th>
            <th width="156">Capital</th>
        </tr>
        <?php for( $f = 0; $f < count($lista); $f++ ) { ?>
        <tr>
            <td><?php echo($lista[$f][0]) ?></td>
            <td><?php echo($lista[$f][1]) ?></td>
        </tr>
        <?php } ?>
    </table>
</body>
```

```
</html>
```

El `for` hace el recorrido por todas las filas; como son solamente dos columnas, se accede directamente utilizando el índice 0 para la primera columna y el índice 1 para la segunda columna.

El resultado es el siguiente:

Conoce América Latina	
País	Capital
Perú	Lima
Bolivia	La Paz
Brazil	Brasilia
Argentina	Buenos Aires
Colombia	Bogotá

Capítulo 6

ARREGLOS ASOCIATIVOS

DEFINICIÓN

Los arreglos asociativos se basan en parejas **clave** → **valor**, quiere decir que el índice puede ser un número o una cadena, y se utilizará como clave para acceder a su valor asociado, tal como se ilustra a continuación:

```
$lista["codigo"] = "ART001";  
$lista["nombre"] = "Impresora";  
$lista["precio"] = 250.00;
```

Ejemplo 80

Este ejemplo crea un arreglo asociativo y luego realiza un listado de todos sus elementos.

Archivo: cap6\ejm80.php

```
<?php  
// Arreglo Asociativo  
$rec["codigo"] = "ART001";  
$rec["nombre"] = "Impresora";  
$rec["precio"] = 250.00;  
?>  
  
<html>  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
    <title>Capítulo 06 - Ejemplo 06</title>  
  </head>  
  <body bgcolor="#D2EBF7">  
    <h3>Arreglo Asociativo</h3>  
    <table width="234" border="1">  
      <tr>  
        <th width="119" scope="col">Atributo</th>  
        <th width="165" scope="col">Valor</th>  
      </tr>  
      <tr>  
        <td>Código</td>
```



```
<td><?php echo( $rec["codigo"] ) ?></td>
</tr>
<tr>
<td>Nombre</td>
<td><?php echo( $rec["nombre"] ) ?></td>
</tr>
<tr>
<td>Precio</td>
<td><?php echo( $rec["precio"] ) ?></td>
</tr>
</table>
</body>
</html>
```

Resultado:

Arreglo Asociativo	
Atributo	Valor
Código	ART001
Nombre	Impresora
Precio	250

USO DE LA FUNCIÓN ARRAY()

La función `array()` también permite crear arreglos asociativos, se debe utilizar la siguiente sintaxis:

```
$lista = array(  
    clave1 => valor1,  
    clave2 => valor2,  
    clave3 => valor3,  
    ...  
    ...  
);
```

Ejemplo 81

Este ejemplo ilustra el uso de un arreglo asociativo con la función `array()`, y luego realiza un listado de sus elementos.

Archivo: `cap6\ejm81.php`

```
<?php  
// Arreglo Asociativo  
$lugares = array(  
    "chiclayo" => "Ciudad de la Amistad",  
    "trujillo" => "Ciudad de la eterna Primavera",  
    "arequipa" => "Ciudad Blanca",  
    "cuzco" => "Capital Arqueológica de América",  
    "huaraz" => "Conocida como la Suiza Peruana"  
);  
?>  
  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
        <title>Capítulo 06 - Ejemplo 07</title>  
    </head>  
    <body bgcolor="#D2EBF7">  
        <h3>Lugares Turísticos Peruanos</h3>  
        <table width="400" border="1">  
            <tr>  
                <th width="111" scope="col">Ciudad</th>  
                <th width="273" scope="col">Descripción</th>
```

```
</tr>
<tr>
  <td>Chiclayo</td>
  <td><?php echo($lugares["chiclayo"]) ?></td>
</tr>
<tr>
  <td>Trujillo</td>
  <td><?php echo($lugares["trujillo"]) ?></td>
</tr>
<tr>
  <td>Arequipa</td>
  <td><?php echo($lugares["arequipa"]) ?></td>
</tr>
<tr>
  <td>Cuzco</td>
  <td><?php echo($lugares["cuzco"]) ?></td>
</tr>
<tr>
  <td>Huaraz</td>
  <td><?php echo($lugares["huaraz"]) ?></td>
</tr>
</table>
</body>
</html>
```

Resultado:

Lugares Turísticos Peruanos	
Ciudad	Descripción
Chiclayo	Ciudad de la Amistad
Trujillo	Ciudad de la eterna Primavera
Arequipa	Ciudad Blanca
Cuzco	Capital Arqueológica de América
Huaraz	Conocida como la Suiza Peruana

USO DE FOREACH()

La estructura de control `foreach` también permite recorrer los elementos de un arreglo asociativo, la sintaxis es:

```
foreach( $arreglo as $clave => $valor ) {  
  
    // instrucciones  
  
}
```

Ejemplo 82

Este ejemplo ilustra el uso de `foreach` para hacer el recorrido por los elementos de un arreglo asociativo.

Archivo: `cap6\ejm82.php`

```
<?php  
$lugares = array(  
    "Túcume" => "Complejo Arqueológico",  
    "Sípan" => "Necrópolis del Señor de Sipan llamada Huaca Rajada",  
    "Monsefú" => "Impresionante pueblo costumbrista",  
    "Motupe" => "Famosa por la Fiesta de la Cruz de Chalpón",  
    "Tinajones" => "Reservorio con hermosos paisajes y una fauna muy variada"  
);  
?>  
  
<html>  
    <head>  
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />  
        <title>Capítulo 06 - Ejemplo 08</title>  
    </head>  
    <body bgcolor="#D2EBF7">  
        <h3>Visita Chiclayo</h3>  
        <table width="400" border="1">  
            <tr>  
                <th scope="col">Lugar</th>  
                <th scope="col">Descripción</th>  
            </tr>  
            <?php foreach ( $lugares as $clave => $dato ) { ?>  
                <tr>
```

```
<td><?php echo($clave) ?></td>  
<td><?php echo($dato) ?></td>  
</tr>  
<?php } ?>  
</table>  
</body>  
</html>
```

Resultado:

Visita Chiclayo	
Lugar	Descripción
Túcume	Complejo Arqueológico
Sípan	Necrópolis del Señor de Sipan llamada Huaca Rajada
Monsefú	Impresionante pueblo costumbrista
Motupe	Famosa por la Fiesta de la Cruz de Chalpón
Tinajones	Reservorio con hermosos paisajes y una fauna muy variada

Capítulo 7

SESIONES

¿QUÉ SON LAS SESIONES?

Las sesiones son un mecanismo basado en cookies que permiten identificar a los usuarios que acceden a un sitio Web, de esta manera almacenar información referente a sus transacciones.

Un ejemplo típico donde se aplican sesiones, son las aplicaciones de comercio electrónico, en estas aplicaciones se debe llevar un registro de los productos que ha agregado a su canasta de compras.

MANEJO DE SESIONES

Función: `sesión_start()`

Sintaxis

```
boolean sesión_start ( void )
```

Valor de Retorno

Siempre devuelve TRUE.

Esta función en caso de que el visitante no tenga una sesión la crea y si ya tiene una sesión continua en ella.

La propagación de una sesión es por cookies, claro que de eso se encarga la función, por ello es necesario llamar a esta función antes de enviar cualquier cabecera HTTP, esto es antes de comenzar a imprimir nuestra página, de lo contrario habrá un error al tratar de crear una cookie.

Una vez iniciada la sesión podremos almacenar información de la sesión en el arreglo **`$_SESSION`**. Este es un arreglo asociativo, además es super-global lo que quiere decir que su alcance se extiende a todo el ámbito de la aplicación y no tendrás que declararlo como global al utilizarlo dentro de funciones.

Ejemplo 83

En este ejemplo se ilustra la manera **como no se debe usar** la función **session_start()**.

Archivo: ejm83.php

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Ejemplo de Sesión</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
  <h3>Ejemplo de Sesión</h3>
  <?php
    session_start();
    echo "Estoy en una sesión";
  ?>
</body>
</html>
```

El resultado que se obtiene es:

Ejemplo de Sesión

Warning: session_start() [[function.session-start](#)]: Cannot send session cookie - headers already sent by (output started at C:\PHP100\cap08\ejm01.php:8) in C:\PHP100\cap08\ejm01.php on line 9

Warning: session_start() [[function.session-start](#)]: Cannot send session cache limiter - headers already sent (output started at C:\PHP100\cap08\ejm01.php:8) in C:\PHP100\cap08\ejm01.php on line 9
Estoy en una sesión

Función: session_id()

Sintaxis

```
string session_id ( [string id])
```

Valor de Retorno

Retorna un cadena con el id de sesión.

Esta función se utiliza para leer o cambiar el id de la sesión actual.

Ejemplo 84

Este ejemplo ilustra la manera como se debe trabajar con sesiones.

Archivo: ejm84.php

```
<?php
    session_start();
    $idSesion = session_id();
?>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Ejemplo de Sesión</title>
    </head>
    <body bgcolor="#D2EBF7" style="font-family: Tahoma;">
        <h3>Ejemplo de Sesión</h3>
        <p>Estoy en una sesión</p>
        <p>El Id de sesión es: <?php echo($idSesion) ?></p>
    </body>
</html>
```

El resultado se ilustra a continuación:

Ejemplo de Sesión

Estoy en una sesión

El Id de sesión es: 62ac3c13c1b8d7184c27f6bf04852499

Arreglo: \$_SESSION

Es una matriz asociativa que contiene las variables de sesión disponibles en la aplicación actual.

Se trata de una variable super-global o global automática. Esto simplemente quiere decir que está disponible en todos los contextos a lo largo de una aplicación.

Ejemplo 85

En el siguiente ejemplo se ilustra el uso del arreglo \$_SESSION.

El usuario ingresa nombres de artículo y sus respectivos precios, cada nombre de artículo se utiliza como clave en el arreglo asociativo \$_SESSION.

Archivo: ejm85.php

```
<?php
session_start();

if( isset( $_POST["enviar"] ) ) {
    $_SESSION[ $_POST["articulo"] ] = $_POST["precio"];
}
if( isset( $_POST["limpiar"] ) ) {
    session_unset();
    session_destroy();
}
$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Variables de Sesión</title>
  </head>
  <body bgcolor="#D2EBF7" style="font-family: Tahoma;">
    <h3>Datos de Artículo</h3>
    <form method="post" action="<?php echo($pagina) ?>">
      <table width="240">
        <tr>
          <td width="76">Artículo</td>
          <td width="152"><input type="text" name="articulo"></td>
        </tr>
        <tr>
          <td>Precio</td>
```

```
<td><input type="text" name="precio"></td>
</tr>
</table>
<p>
  <input name="enviar" type="submit" id="enviar" value="Enviar">
  <input type="submit" name="limpiar" id="limpiar" value="Limpiar">
</p>
</form>
<?php if( count($_SESSION) > 0 ) { ?>
<table border='1' width='200'>
  <tr>
    <th>Artículo</th>
    <th>Precio</th>
  </tr>
  <?php foreach ( $_SESSION as $clave => $valor ) { ?>
  <tr>
    <td><?php echo($clave) ?></td>
    <td><?php echo($valor) ?></td>
  </tr>
  <?php } ?>
</table>
<?php } ?>
</body>
</html>
```

El resultado se ilustra a continuación:

Datos de Artículo

Artículo

Precio

Artículo	Precio
Televisor	580
Impresora	278
Disco Duro	340

Función: session_unset()

Sintaxis

```
void session_unset ( void )
```

Valor de Retorno

No retorna ningún valor.

Esta función elimina y libera el espacio ocupado por todas las variables de sesión actualmente registradas.

Función: session_destroy()

Sintaxis

```
bool session_destroy ( void )
```

Valor de Retorno

Retorna un valor lógico que indica el estado de su ejecución.

Esta función destruye todos los datos asociados con la sesión actual. No destruye ninguna de las variables globales asociadas a la sesión ni la cookie.

Retorna TRUE si se ha destruido la sesión correctamente y FALSE en caso de haber algún problema.

Ejemplo 86

En este ejemplo se utilizará dos programas, los cuales se detallan a continuación:

Programa	Descripción
Ejm86.php	Este programa es prácticamente igual al programa ejm03.php, salvo por el link hacia el programa ejm86_delete.php .
Ejm86_delete.php	Este programa se encarga de eliminar la sesión actual, y luego redirecciona la ejecución al programa ejm86.php .

Archivo: ejm86.php

```
<?php
session_start();

if( isset( $_POST["enviar"] ) ) {
    $_SESSION[ $_POST["articulo"] ] = $_POST["precio"];
}
$pagina = "http://" . $_SERVER['HTTP_HOST'] . $_SERVER['SCRIPT_NAME'];
?>

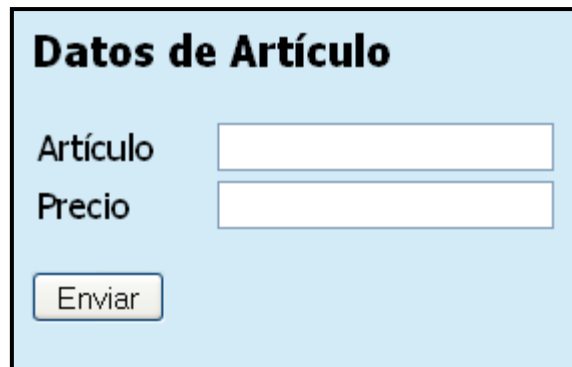
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Variables de Sesión</title>
  </head>
  <body bgcolor="#D2EBF7" style="font-family: Tahoma;">
    <h3>Datos de Artículo</h3>
    <form method="post" action="<?php echo($pagina) ?>">
      <table width="240">
        <tr>
          <td width="76">Artículo</td>
          <td width="152"><input type="text" name="articulo"></td>
        </tr>
        <tr>
          <td>Precio</td>
          <td><input type="text" name="precio"></td>
        </tr>
      </table>
      <p>
        <input name="enviar" type="submit" id="enviar" value="Enviar">
      </p>
    </form>
    <?php if( count($_SESSION) > 0 ) { ?>
    <table border='1' width='200'>
      <tr>
        <th>Artículo</th>
        <th>Precio</th>
      </tr>
      <?php foreach ( $_SESSION as $clave => $valor ) { ?>
      <tr>
        <td><?php echo($clave) ?></td>
        <td><?php echo($valor) ?></td>
      </tr>
    </table>
  </body>
</html>
```

```
<?php } ?>
</table>
<a href="ejm04_delete.php">Eliminar Sesión</a>
<?php } ?>
</body>
</html>
```

Archivo: ejm86_delete.php

```
<?php
session_start();
session_unset();
session_destroy();
header("location: ejm04.php");
?>
```

Cuando ejecuta el programa **ejm04.php**, obtiene la siguiente interfaz:

El formulario tiene un fondo azul claro y un título "Datos de Artículo" en negrita. Contiene dos etiquetas "Artículo" y "Precio" con sus respectivos campos de entrada de texto. Debajo de estos campos hay un botón rectangular con el texto "Enviar".

Datos de Artículo	
Artículo	<input type="text"/>
Precio	<input type="text"/>
<input type="button" value="Enviar"/>	

Luego de ingresar algunos datos obtenemos la siguiente interfaz:

Datos de Artículo

Artículo

Precio

Enviar

Artículo	Precio
Coca Cola	3
Combo Familiar	35

[Eliminar Sesión](#)

El enlace **Eliminar Sesión** ejecuta el programa **ejm04_delete.php** que se encarga de eliminar la sesión y todas sus variables, luego redirecciona el control al programa **ejm04.php**, y nuevamente aparece el formulario, pero ya no los artículos, porque han sido eliminados todos los datos de la sesión.

EJEMPLO APLICATIVO

En este ejemplo se ilustra la implementación de un carrito de compras básico, esta demostración está compuesta por los siguientes programas:

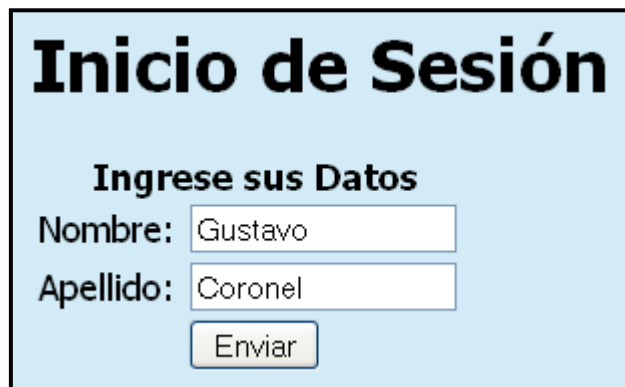
Programa	Descripción
inicio.html	Este programa muestra el formulario para que el usuario ingrese sus datos de inicio de sesión, y los envía al programa inicio.php .
inicio.php	Este programa procesa los datos enviados por el programa inicio.html , específicamente inicia la sesión y crea las variables de sesión respectiva.
carrito.php	Este programa muestra el contenido del carrito, y enlaces hacia los programas agregararticulo.php y cerrarsesion.php .
agregaarticulo.php	Este programa muestra un formulario para agregar un nuevo artículo al carrito de compras, luego redirecciona el control al programa carrito.php .
cerrarsesion.php	Este programa cierra la sesión y luego redirecciona el control al programa inicio.html .

Archivo: inicio.html

```
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>Inicio de Sesión</title>
</head>
<body bgcolor="#D2EBF7" style="font-family: Tahoma;">
  <h1>Inicio de Sesión</h1>
  <form method="post" action="inicio.php" >
    <table width="200">
      <caption><strong>Ingrese sus Datos</strong></caption>
      <tr>
        <td>Nombre:</td>
        <td>
          <input type="text" name="nombre" size="15" maxlength="15">
        </td>
      </tr>
    </table>
  </form>
</body>
</html>
```

```
<tr>
  <td>Apellido: </td>
  <td>
    <input type="text" name="apellido" size="15" maxlength="15">
  </td>
</tr>
<tr>
  <td colspan="2" align="center">
    <input name="enviar" type="submit" value="Enviar" id="enviar">
  </td>
</tr>
</table>
</form>
</body>
</html>
```

Este programa permite al usuario identificarse, presenta la siguiente interfaz para que ingrese su nombre y apellido:



Archivo: inicio.php

```
<?php
session_start();
if( ! isset( $_POST["nombre"] ) || ! isset( $_POST["apellido"] ) ) {
  $destino = "inicio.html";
} else {
  $_SESSION["nombre"] = $_POST["nombre"];
  $_SESSION["apellido"] = $_POST["apellido"];
  $_SESSION["carrito"] = array();
  $destino = "carrito.php";
}
header( "location: $destino" );
?>
```


Este programa recibe los datos ingresados en el formulario del programa **inicio.html**, inicializa la sesión y luego redirecciona el control hacia el programa **carrito.php**.

Archivo: carrito.php

```
<?php
session_start();
if( ! isset($_SESSION["nombre"]) ){
    header( "location: inicio.html" );
    return;
}
$carritoVacio = (count($_SESSION["carrito"]) == 0);
$nombre = "{$_SESSION["nombre"]} {$_SESSION["apellido"]}";
?>

<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <title>Inicio de Sesión</title>
    </head>
    <body bgcolor="#D2EBF7" style="font-family: Tahoma;">
        <h2>Usuario: <?php echo( $nombre ) ?></h2>
        <table width="358">
            <tr>
                <td width="170" align="center">
                    <a href="agregararticulo.php">Agregar Artículo</a>
                </td>
                <td width="176" align="center">
                    <a href="cerrarsesion.php">Cerrar Sesión</a>
                </td>
            </tr>
        </table>
        <?php if( $carritoVacio ) { ?>
        <h3>Carrito Vacío</h3>
        <?php } else { ?>
        <table width='447' border='1'>
            <tr>
                <th width="188">Artículo</th>
                <th width="73">Precio</th>
                <th width="65">Cant.</th>
                <th width="93">Subtotal</th>
            </tr>
            <?php foreach ( $_SESSION["carrito"] as $rec ) { ?>
```

```
<tr>
  <td><?php echo($rec["articulo"]) ?></td>
  <td><?php echo($rec["precio"]) ?></td>
  <td><?php echo($rec["cantidad"]) ?></td>
  <td><?php echo($rec["subtotal"]) ?></td>
</tr>
<?php } ?>
</table>
<?php } ?>
</body>
</html>
```

Después de agregar varios artículos este programa tendrá el siguiente resultado:

Usuario: Gustavo Coronel			
Agregar Artículo	Cerrar Sesión		
Articulo	Precio	Cant.	Subtotal
Televisor	589	4	2356
Teclado	30	3	90
Impresora	230	6	1380

Archivo: agregararticulo.php

```
<?php
session_start();
if( ! isset($_SESSION["nombre"]) ){
  header( "location: inicio.html" );
  return;
}
if( isset( $_POST["enviar"] ) ) {
  $articulo = $_POST["articulo"];
  $rec["articulo"] = $_POST["articulo"];
  $rec["precio"] = $_POST["precio"];
  $rec["cantidad"] = $_POST["cantidad"];
  $rec["subtotal"] = $rec["precio"] * $rec["cantidad"];
  $_SESSION["carrito"][$articulo] = $rec;
  header( "location: carrito.php" );
  return;
}
```

```
}
$nombre = "{$_SESSION["nombre"]} {$_SESSION["apellido"]}";
?>

<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Inicio de Sesión</title>
  </head>
  <body bgcolor="#D2EBF7" style="font-family: Tahoma;">
    <h2>Usuario: <?php echo($nombre) ?></h2>
    <form name="form1" method="post" action="agregararticulo.php">
      <table width="282">
        <tr>
          <td width="74">Artículo</td>
          <td width="182">
            <input type="text" name="articulo" size="30" maxlength="30">
          </td>
        </tr>
        <tr>
          <td>Precio</td>
          <td>
            <input name="precio" type="text" id="precio"
              size="10" maxlength="5">
          </td>
        </tr>
        <tr>
          <td>Cantidad</td>
          <td>
            <input type="text" name="cantidad" size="10" maxlength="3">
          </td>
        </tr>
        <tr>
          <td colspan="2" align="center">
            <input type="submit" name="enviar" value="Enviar" id="enviar">
          </td>
        </tr>
      </table>
    </form>
  </body>
</html>
```

Inicialmente este programa presenta el siguiente formulario:

Usuario: Gustavo Coronel

Artículo

Precio

Cantidad

Enviar

Luego de agregar el nombre del artículo, el precio y la cantidad, agrega estos datos al carrito y luego se redirecciona el control al programa **carrito.php**.

Archivo: **cerrarsesion.php**

```
<?php
    session_start();
    session_unset();
    session_destroy();
    header( "location: inicio.html" );
?>
```

Después de crear todos los programas puede empezar a interactuar con ellos, debe iniciar la ejecución con el programa **inicio.html**.