1.- Inserción de PHP en HTML

EL PHP va dentro dentro de la página HTML (o página PHP, según se vea) y para insertarlo dentro de utilizan, entre otras cosas, las instrucciones de procesado (<? ... ?>).

- <? .... ?> Sólo si se activa la función**short_tags()** o la bandera de configuración*short_open_tag.*
- <?php .... ?>
- <script languaje="php"> .... </script>
- <% .... %> Sólo si se activan los tags para ficheros *'asp'* con la bandera de configuración*asp_tags.*

Ejemplos de PHP en HTML:

```
<html>
<head> <title>Ejemplo 1 </title></head>
<body>
 <h1> Ejemplo de PHP </h1>

 <? echo ("esta es la más simple, una instrucción de procesado SGML\n"); ?>
<br>

 <?php print "si quiere servir documentos XML, se puede hacer esto\n"; ?>
<br>

 <script language="php">
        printf ("a algunos editores -como FrontPage- no les
              gustan las intrucciones de procesado");
 </script>
<br>

 <% echo ("Puedes también usar etiquetas tipo ASP"); %>
<br>

<br>
</body>
</html>
```

En el ejemplo anterior se puede observar que **echo**, **printf** y**print** sirven para lo mismo para escribir en la página HTML que genera PHP como resultado .

2.- Separación de instrucciones

Las instrucciones se separan con *';'*, en el caso de ser la última instrucción no es necesario el punto y coma.

3.- Comentarios

Los comentarios en PHP pueden ser:

- Como en C o C++, /*...*/ ó //
- Otro tipo de comentario de una línea es #, que comentará la línea en la que aparezca pero sólo hasta el tag *?>* que cierra el código php.

4.- Tipos de Datos

Los tipos de cada variable en PHP no están tan claros como en C. El intérprete asigna el tipo de una variable según el uso que se esté haciendo de ella. Para asignar un tipo fijo a una variable se utiliza la función **settype()**. Los tipos son:

- Enteros
- Flotantes
- String
- Arrays
- Objetos
- Variables variables

Respecto al tipo entero y flotante, no hay mucho que decir, así que detallaremos sólo los tipos String, Arrays y Objetos.

Ejemplos de Tipos de datos simples de PHP:

```
<html>
<head> <title>Ejemplo 2 </title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php

 #Enteros
 $a = 1234; # número decimal
 $a = -123; # un número negativo
 $a = 0123; # número octal (equivalente al 83 decimal)
 $a = 0x12; /* número hexadecimal (equivalente al 18 decimal) */

 //Flotantes o reales
 $b = 1.234; $b = 1.2e3;

 //Escribimos algo
 print "\n La a= $a y la b= $b <br>\n";
?>

</body>
</html>
```

En el ejemplo anterior puede verse que se han usado distintos tipos de comentarios

4.1. -String

Las cadenas pueden estar delimitadas por *"* o *'*. Si la cadena está delimitada por comillas dobles, cualquier variable incluida dentro de ella será sustituida por su valor (ver y ejecutar el ejemplo anterior). Para especificar el carácter *"* se escapará con el carácter backslash(\ ) Otra forma de delimitar una cadena es utilizando la sintaxis de documentos incrustado "<<<" Ejemplo:

```
  $variable = <<< EOD
  Ejemplo de cadena
  que ocupa
  varias líneas
  EOD;
```

La marca de final de un docuemento incrustado (en este caso EOD) debe estar en la primera columan del documento.

Las operaciones con cadenas son exactamente igual que en PERL. Por ejemplo, con **strlen** se ve el tamaño de una cadena y con el punto (**.**) se concatenan cadenas.

Operaciones con cadenas

```
<html>
<head> <title>Ejemplo 3 </title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php
 /* Asignando una cadena. */
 $str = "Esto es una cadena";

 /* Añadiendo a la cadena. */
 $str = $str . " con algo más de texto";

 /* Otra forma de añadir, incluye un carácter de nueva línea  */
 $str .= " Y un carácter de nueva línea al final.\n";
 print "$str <br>\n";
```

```
 /* Esta cadena terminará siendo '<p>Número: 9</p>' */
 $num = 9;
 $str = "<p>Número: $num</p>";
 print "$str <br>\n";

 /* Esta será '<p>Número: $num</p>' */
 $num = 9;
 $str = '<p>Número: $num</p>';
 print "$str <br>\n";

 /* Obtener el primer carácter de una cadena  como una vector*/
 $str = 'Esto es una prueba.';
 $first = $str[0];
 print "$str 0->$first <br>\n";

 /* Obtener el último carácter de una cadena. */
 $str = 'Esto es aún una prueba.';
 $last = $str[strlen($str)-1];
 print "$str last->$last <br>\n";
 ?>

</body>
</html>
```

Para hacer conversión de cadenas a otros tipos de datos hay que tener en cuenta una cadena se evalúa como un valor numérico, el valor resultante y el tipo se determinan como sigue. La cadena se evaluará como un doble si contiene cualquiera de los caracteres '.', 'e', o 'E'. En caso contrario, se evaluará como un entero. El valor viene dado por la porción inicial de la cadena. Si la cadena comienza con datos de valor numérico, este será el valor usado. En caso contrario, el valor será 0 (cero). Cuando la primera expresión es una cadena, el tipo de la variable dependerá de la segunda expresión.

Ejemplos de tipos de datos

```
<html>
<head> <title>Ejemplo 4</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php

 $foo = 1 + "10.5";              // $foo es doble (11.5)
 print "$foo <br>\n";
 $foo = 1 + "-1.3e3";            // $foo es doble (-1299)
 print "$foo <br>\n";
 $foo = 1 + "bob-1.3e3";         // $foo es entero (1)
 print "$foo <br>\n";
 $foo = 1 + "bob3";              // $foo es entero (1)
 print "$foo <br>\n";
 $foo = 1 + "10 Cerditos";     // $foo es entero (11)
 print "$foo <br>\n";
 $foo = 1 + "10 Cerditos"; // $foo es entero (11)
 print "$foo <br>\n";
 $foo = "10.0 cerdos " + 1;        // $foo es entero (11)
 print "$foo <br>\n";
 $foo = "10.0 cerdos " + 1.0;      // $foo es doble (11)
 print "$foo <br>\n";

?>

</body>
</html>
```

4.2.-Arrays

Los Arrays en PHP se pueden utlizar tanto como Arrays indexados (vectores) o como Arrays

asociativos (tablas hash). Para PHP, no existen ninguna diferencia arrays indexados unidimensionales y arrays asociativos. Las funciones que se utilizan para crear Arrays son **list()** o **array()** , o se puede asignar el valor de cada elemento del array de manera explícita. En el caso de que no se especifique el índice en un array, el elemento que se asigna se añade al final.

Ejemplos de Arrays

```
<html>
<head> <title>Ejemplo 5</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php

 #forma explicita
 $a[0] = "abc";
 $a[1] = "def";
 $b["foo"] = 13;

 #Añadiendo valores al array
 $a[] = "hola"; // $a[2] == "hola"
 $a[] = "mundo"; // $a[3] == "mundo"

 #mostramos los resultados
 print "a= $a[0] , $a[1] , $a[2] , $a[3] <br>\n";
 print "b[foo]=".$b["foo"]."<br>\n";

?>

</body>
</html>
```

Los arrays se pueden ordenar usando las funciones asort(),arsort(), ksort(), rsort(), sort(), uasort(), usort(), y uksort()dependiendo del tipo de ordenación que se desee.

Se puede contar el número de elementos de un array usando la función count().

Se puede recorrer un array usando las funciones next() yprev(). Otra forma habitual de recorrer un array es usando la función each().

Los arrays multidimensionales son bastante simples, para cada dimensión array, se puede añadir otro valor [clave] al final. Los indices de un array multidimensional pueden ser tanto numéricos como asociativos.

Arrays multidimensionales

```
 $a[1]      = $f;          # ejemplos de una sola dimensión
 $a["foo"]  = $f;

 $a[1][0]     = $f;        # bidimensional
 $a["foo"][2] = $f;        # (se pueden mezclar índices numéricos y asociativos)
 $a[3]["bar"] = $f;        # (se pueden mezclar índices numéricos y asociativos)

 $a["foo"][4]["bar"][0] = $f;   # tetradimensional!
```

Los arrays se puede rellenar también usando =>

```
 # Ejemplo 1:
 $a["color"]     = "rojo";
 $a["sabor"]     = "dulce";
 $a["forma"]     = "redondeada";
 $a["nombre"]    = "manzana";
 $a[3]           = 4;

 # Ejemplo 2:
```

```
$a = array(
     "color" => "rojo",
     "sabor" => "dulce",
     "forma" => "redondeada",
     "nombre"  => "manzana",
     3       => 4
);
```

4.3.-Objetos

Para inicializar un objeto se utiliza el método *new* , y para acceder a cada uno de sus métodos se utiliza el operador -> .

```
class nada {
     function haz_nada () {
         echo "No estoy haciendo nada.";
     }
}

$miclase = new nada;
$miclase->haz_nada();
```

Posteriormente veremos como trabajar con objetos en PHP con más profundidad.

4.4.-Conversión de Tipos de datos

Una variable en PHP, define su tipo según el contenido y el contexto en el que se utilice, es decir, si se asigna una cadena a una variable, el tipo de esa variable será *string* . Si a esa misma variable se el asigna un número, el tipo cambiará a *entero* .

Para asegurarte de que una variable es del tipo adecuado se utiliza la función **settype()** . Para obtener el tipo de una variable se utiliza la función **gettype()** .

También es posible utilizar el mecanismo del *casting* tal y como se utiliza en C.

Ejemplos de Castings

```
<html>
<head> <title>Ejemplo 6</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php

 $foo = 10;   // $foo es un entero
 $bar = (double) $foo;   // $bar es un doble

 #Mostramos resultados
 print "bar=$bar , foo=$foo <br>\n";

?>

</body>
</html>
```

Los tipos de casting permitidos son:

- (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)

5.- Variables

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas. Las variables se asignan normalmente por valor, pero desde PHP4, también se asignan por referencia usando el símbolo **&**

Variables por valor y referencia

```
<html>
<head> <title>Ejemplo 7</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php
 $foo = 'Bob';              // Asigna el valor 'Bob' a $foo
 $bar = &$foo;              // Referencia $foo vía $bar.
 $bar = "Mi nombre es $bar";  // Modifica $bar...
 echo $foo." <br>\n";                 // $foo también se modifica.
 echo $bar." <br>\n";
?>

</body>
</html>
```

Algo importante a tener en cuenta es que sólo las variables con nombre pueden ser asignadas por referencia.

5.2.- Variables predefinidas

En PHP cada vez que se ejecuta un script, existen variables que se crean y que nos pueden informar del entorno en el que se está ejecutando dicho script.

Para obtener una lista de todas estas variables predefinidas se puede utilizar la funcion **PHPinfo()**.

De todas estas variables, algunas se crean dependiendo del servidor que se esté utilizando y otras son propias de PHP.

Si se tratara de un servidor Apache, la lista de variables es:

- GATEWAY_INTERFACE:
- SERVER_NAME
- SERVER_SOFTWARE
- SERVER_PROTOCOL
- REQUEST_METHOD
- QUERY_STRING
- DOCUMENT_ROOT
- HTTP_ACCEPT
- HTTP_ACCEPT_CHARSET
- HTTP_ENCODING
- HTTP_ACCEPT_LANGUAJE
- HTTP_CONNECTION
- HTTP_HOST
- HTTP_REFERER
- HTTP_USER_AGENT
- REMOTE_ADDR
- REMOTE_PORT
- SCRIPT_FILENAME
- SERVER_ADMIN
- SERVER_PORT
- SERVER_SIGNATURE
- PATH_TANSLATED
- SCRIPT_NAME
- REQUEST_URL

las variables creadas por el propio PHP son:

- argv
- argc
- PHP_SELF

- HTTP_COOKIE_VARS
- HTTP_GET_VARS
- HTTP_POST_VARS

Nota: Esta lista no es exhaustiva ni pretende serlo. Simplemente es una guía de qué tipo de variables predefinidas se puede esperar tener disponibles en un script PHP.

5.3.- Ámbito de una Varible

El ámbito de una variable en PHP es exactamente igual que en C o en Perl tomando siempre en cuenta los ficheros incluidos al principio de cada programa.

La única diferencia se encuentra en las variables globales, que tienen que ser expresamente definidas dentro de las funciones.

Variables globales

```
<html>
<head> <title>Ejemplo 8</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php
 $a = 1;
 $b = 2;

 Function Sum () {
     global $a, $b;

     $b = $a + $b;
 }

 Sum ();
 echo $b;

 ?>

</body>
</html>
```

5.4. Varibles variables

PHP permite un mecanismo para mantener variables con un nombre no fijo.

Por ejemplo:

```
   $a = "hola";
   $$a = "mundo";
```

El ejemplo anterior, define dos variables, una denominada $a que contiene el valor "hola" y otra que se llama $hola que contiene el valor "mundo"

Para acceder al valor de una variable, se accede con:

```
                                    echo "$a ${$a}";
```

La sentencia anterior provocará la salida "hola mundo".

Algo que se debe tener en cuenta cuando se utilizan variables, es que hay que resolver la ambiguedad que se crea al utilizar arrays de variables de este tipo. Por ejemplo *$$a[1]*provoca una ambiguedad para el intérprete, puesto que no sabe si se desea utilizar la variable denominada $a[1] o utilizar la variables $a indexándola en su primer valor. Para esto se utiliza una sintaxis especial que sería *${$a[1]}* o *${$a}[1]* según se desee una opción u otra.

5.5.- Variables de los formularios HTML

Cuando existe un formulario en HTML, inmediatamente después de ser enviado, dentro del ámbito PHP se crean automáticamente una variable por cada uno de los objetos que contiene el formulario.

Por ejemplo, consideremos el siguiente formulario:

Formulario simple

```
<html>
<head> <title>Ejemplo 9</title></head>
<body>
 <h1> Ejemplo de Formulario 1 </h1>

<p>
Dame tu nombre !!!

<form action="ej10.php" method="post">
     Nombre: <input type="text" name="nombre">

     <input type="submit">
</form>

</body>
</html>
```

Cuando es enviado, PHP creará la variable $nombre, que contendrá lo que sea que se introdujo en el campo Nombre:: del formulario.

Procesado de Formulario siple

```
<html>
<head> <title>Ejemplo 10</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php
 print "<h2>Hola $nombre </h2>\n";

?>

</body>
</html>
```

PHP también maneja arrays en el contexto de variables de formularios, pero sólo en una dimensión. Se puede, por ejemplo, agrupar juntas variables relacionadas, o usar esta característica para recuperar valores de un campo select input múltiple:

Formulario complejos

```
<html>
<head> <title>Ejemplo 11</title></head>
<body>
 <h1> Ejemplo de Formulario 2 </h1>


<form action="ej12.php" method="post">
     Nombre: <input type="text" name="personal[name]">

     E-mail: <input type="text" name="personal[email]">

     Cerveza: <br>
     <select multiple name="beer[]">
         <option value="warthog">Warthog
         <option value="guinness">Guinness
         <option value="stuttgarter">Stuttgarter Schwabenbräu
     </select>
     <input type="submit">
 </form>
```

```
</body>
</html>
```

Variables de formularios complejos

```
<html>
<head> <title>Ejemplo 12</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php

 print "<h2>Hola $personal[name] , ";
 print "tu email es $personal[email] y ";
 print "te gusta la cerveza $beer[0] </h2>\n";

?>

</body>
</html>
```

Si la posibilidad de PHP de track_vars está activada (se hace en la configurtación previa a la compilación), las variables enviadas con los métodos POST o GET también se encontrarán en los arrays asociativos globales *$HTTP_POST_VARS* y*$HTTP_GET_VARS*.

5.6.- Cookies HTML

Las cookies son un mecanismo para almacenar datos en el navegador y así rastrear o identificar a usuarios que vuelven.

La función *SetCookie()* es una función PHP para asignar Cookies a un ordenador cliente. Esta función se debe llamar siempre antes de comenzar a crear la página puesto que debe formar parte de la cabecera de HTML.

Cualquier Cookie que se envía a un cliente, se convierte dentro de PHP en una variable.

Ejemplos de cookies con PHP

```
<?php
 $cuenta++;
 SetCookie ("cuenta", $cuenta, time()+3600);
 SetCookie ("visita[$cuenta]", "Visita número $cuenta", time()+3600);
?>

<html>
<head> <title>Ejemplo 13</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php
 print "cuenta: $cuenta <br>\n";

 print "Número de Visitas:".count($visita)."  <br>\n";

 for ($i=0; $i < count ($visita) ; $i++)
 {
   print "Visita $i:".$visita[$i]." <br>\n";
 }

?>

</body>
</html>
```

Este ejemplo mantiene dos Cookies en el cliente. La primera mantiene el contador *cuenta* y la segunda contiene una lista de los comentarios de cada una de las veces que se ha actualizado la

cookie, *visita.*

Más adelante veremos con más profundidad el uso de las cookies.

5.7.- Variables de entorno

Las variables de entorno, tales como $HOME, para entornos Linux, se pueden utilizar desde PHP.

Para asegurarnos podemos usar la función getenv()

También se puede asignar un valor a una variable de entorno con la función putenv()

6.- Constantes

Las constantes en PHP tienen que ser definidas por la función *define()* y además no pueden ser redefinidas con otro valor.

Además, existen una serie de variables predefinidas denominadas:

- _FILE_: Fichero que se está procesando.
- _LINE_: Línea del fichero que se está procesando
- _PHP_VERSION: Versión de PHP.
- PHP_OS: Sistema operativo del cliente.
- TRUE: Verdadero.
- FALSE: Falso.
- E_ERROR: Error sin recuperación.
- E_WARNING: Error recuperable.
- E_PARSE: Error no recuperable (sintaxis).
- E_NOTICE: Puede Tratarse de un error o no. Normalmente permite continuar la ejecución.
  Ejemplo:

Todas las constantes que empiezan por "E_"se utilizan normalmente con la función *error_reporting()*.

Ejemplos de constantes

```
<html>
<head> <title>Ejemplo 14</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php
define("CONSTANTE", "hello world.");
echo CONSTANTE;

?>

</body>
</html>
```

7.- Expresiones y operadores

En PHP una expresión es cualquier cosa que pueda contener un valor. Las expresiones más simples son las variables y las constantes y otras más complicadas serán las funciones, puesto que cada función devuelve un valor al ser invocada, es decir, contiene un valor, por lo tanto, es una expresión.

Todas las expresiones en PHP son exactamente igual que en C. Los operadores abreviados, los incrementos, etc, son exactamente iguales. Incluso existen otros operadores adicionales como el operador "." que concatena valores de variables, o el operador "===" denominado operador de identidad que devolverá verdadero si las expresiones a ambos lados del operador contienen el mismo valor y a la vez son del mismo tipo. Por último, el operador "@" sirve para el control de errores. Para poder ver como funciona el operador @, veamos un ejemplo:

```
  <?php
  $res = @mysql_query("select nombre from clientes")
or die   ("Error en la selección, '$php_errormsg'");
  ?>
```

Este ejemplo, utiliza el operador @ en la llamada a*mysql_query* y en el caso de dar un error, se salvará el mensaje devuelto en una variable denominada *php_errormsg*. Esta variable contendra el mensaje de error de cada sentencia y si ocurre otro error posterior, se machaca el valor con la nueva cadena.

PHP mantiene también los operadores " ' " que sirven para ejecutar un comando del sistema tal y como hace la función*system()*.

En PHP existen dos operadores *and* y dos operadores *or*que son: 'and', '&&' y 'or', '||' respectivamente, que se diferencian en la precedencia de cada uno.

La tabla que nos puede resumir la precedencia de cada uno de los operadores es:

Asocitividad	Operadores
Izquierda	,
Izquierda	or
Izquierda	xor
Izquierda	and
Derecha	print
Izquierda	= += -* *= /= .= %= &= |= ^= ~= <<= >>=
Izquierda	?:
Izquierda	||
Izquierda	&&
Izquierda	|
Izquierda	^
Izquierda	&
No posee	== != ===
No posee	< <= > >=
Izquierda	>> <<
Izquierda	+ - .
Izquierda	* / %
Derecha	! ~ ++ -- (int) (double) (string) (array) (object) @
Derecha	[
No posee	new

Ejemplos de expresiones

```
<html>
<head> <title>Ejemplo 15</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php

 function double($i) {
     return $i*2;
 }


 $b = $a = 5;        /* asignar el valor cinco a las variables $a y $b */
 $c = $a++;          /* postincremento, asignar el valor original de $a (5) a $c */
 $e = $d = ++$b;     /* preincremento, asignar el valor incrementado de $b (6) a
                        $d y a $e */

 /* en este punto, tanto $d como $e son iguales a 6 */
 $f = double($d++);  /* asignar el doble del valor de $d antes
                        del incremento, 2*6 = 12 a $f */
 $g = double(++$e);  /* asignar el doble del valor de $e después
                        del incremento, 2*7 = 14 a $g */
 $h = $g += 10;      /* primero, $g es incrementado en 10 y termina valiendo 24,
                        después el valor de la asignación (24) se asigna a $h,
```

```
                        y $h también acaba valiendo 24. */

 #Operador de ejecución
 $output = `ls -al`;
 echo "<pre>$output</pre><br>";


 echo "<h3>Postincremento</h3>";
 $a = 5;
 echo "Debería ser 5: " . $a++ . "<br>\n";
 echo "Debería ser 6: " . $a . "<br>\n";

 echo "<h3>Preincremento</h3>";
 $a = 5;
 echo "Debería ser 6: " . ++$a . "<br>\n";
 echo "Debería ser 6: " . $a . "<br>\n";

 echo "<h3>Postdecremento</h3>";
 $a = 5;
 echo "Debería ser 5: " . $a-- . "<br>\n";
 echo "Debería ser 4: " . $a . "<br>\n";

 echo "<h3>Predecremento</h3>";
 $a = 5;
 echo "Debería ser 4: " . --$a . "<br>\n";
 echo "Debería ser 4: " . $a . "<br>\n";
?>

</body>
</html>
```

8.- Estructuras de Control

Además de la sintaxis normal (parecida al Perl o al C), PHP ofrece una sintaxis altenativa para alguna de sus estructuras de control; a saber, if, while, for, y switch. En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por dos puntos (:) y cerrar-llave por endif;, endwhile;, endfor;, or endswitch;, respectivamente.

Ejemplo de sintaxis alternativa para el if

```
<html>
<head> <title>Ejemplo 16</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php


$a=8;
$b=6;

#Primer if
if ($a > $b) {
      print "a es mayor que b<br>";
      $b = $a;
  }

#if alternativo
if ($a > $b):
  print "A es mayor que B<br>";
endif;

#Segundo if (con else y elseif )
if ($a > $b) {
      print "a es mayor que b<br>";
  } elseif ($a == $b) {
```

```
      print "a es igual que b<br>";
  } else {
      print "b es mayor que a<br>";
  }

#Segundo if alternativo
 if ($a > $b):
      print "A es mayor que B<br>";
      print "...";
 elseif ($a == $b):
      print "A es igual a B<br>";
      print "!!!";
 else:
      print "B es mayor que A<br>";
  endif;
?>

</body>
</html>
```

La mejor forma de resumir cada una de las opciones que ofrece PHP para las estructuras de control es mediante una tabla:

Estructura	Alternativa
If, if else, if elseif	if: endif;
while	while: endwhile;
for	for: endfor;
do.. while	-
foreach(array as $value) foreach(array as $key=>$value)	-
switch	switch: endswitch;
continue	-
break	-
require()(Necesitan estar dentro de tags PHP)	-
include()(Necesitan estar dentro de tags PHP)	-

La sentencia **require()** se sustituye a sí misma con el archivo especificado, tal y como funciona la directiva #include de C. La sentencia **include()** incluye y evalúa el archivo especificado.

A diferencia de **include()**, **require()** siempre leerá el archivo referenciado, incluso si la línea en que está no se ejecuta nunca. Si se quiere incluir condicionalmente un archivo, se usa**include()**. La sentencia conditional no afecta a **require()**. No obstante, si la línea en la cual aparece el **require()** no se ejecuta, tampoco se ejecutará el código del archivo referenciado.

De forma similar, las estructuras de bucle no afectan la conducta de **require()**.. Aunque el código contenido en el archivo referenciado está todavía sujeto al bucle, el propio**require()** sólo ocurre una vez. Esto significa que no se puede poner una sentencia **require()** dentro de una estructura de bucle y esperar que incluya el contenido de un archivo distinto en cada iteración. Para hacer esto, usa una sentencia**include()**. Así, require, reemplaza su llamada por el contenido del fichero que requiere, e include, incluye y evalua el fichero especificado.

Ejemplo de include 1

```
<?php
print "Hola Caracola<br>\n";
?>
```

Ejemplo de include 2

```
<html>
<head> <title>Ejemplo 18</title></head>
<body>
 <h1> Ejemplo de PHP </h1>
```

```
<?php

include( 'ej17.php' );

?>

</body>
</html>
```

9.-. Funciones

9.1.- Funciones definidas por el usuario

Un ejemplo puede ser:

```
  function foo($arg1, $arg2, ..., $argN)
    {
      echo "Función ejemplo"
      return $value;
    }
```

Dentro de una función puede aparecer cualquier cosa, incluso otra función o definiciones de clase.

Respecto al paso de argumentos, son siempre pasados por valor y para pasarlos por referencia hay que indicarlo y se puede hacer de dos formas diferentes, en la definición de la función, anteponiendo el símbolo & al argumento que corresponda, en este caso la llamada será igual que la llamada a una función normal, o manteniendo la definición de la función normal y anteponer un & delante del argumento que corresponda en la llamada a la función.

Ejemplo de parámetros de funciones

```
<html>
<head> <title>Ejemplo 19</title></head>
<body>
 <h1> Ejemplo de PHP </h1>

<?php

#Definimos la función con parametros por referencia
function suma1 (&$a, &$b)
{
  $c=$a+$b;
  return $c;
}

#Definimos la función con parametros por valor
function suma2 ($a, $b)
{
  $c=$a+$b;
  return $c;
}

$a=2; $b=3; $suma;
#Llamamos la función 1 por referencia (no puede ser de otra forma)
print $suma=suma1($a,$b);

#Llamamos la función 2 por referencia
print $suma=suma1(&$a,&$b);

#Llamamos la función 2 por valor
print $suma=suma1($a,$b);

?>

</body>
```

```
</html>
```

PHP permite el mecanismo de argumentos por defecto. Un ejemplo de esta caracteristica es:

```
function hacerCafe($tipo="capuchino")
  {
    return "he hecho un café $tipo\n";
  }
```

En la llamada a esta función se obtendrá una frase u otra según se llame:

```
echo hacerCafe();
echo hacerCafe("expreso");
```

En el caso de tratarse de una función con argumentos por defecto y argumentos normales, los argumentos por defecto deberán estar agrupados al final de la lista de argumentos.

En PHP4 el número de argumentos de una función definida por el usuario, puede ser variable, se utilizan las funciones**func_num_args()**, **func_get_arg()** y **func_get_args()**.

9.2- Valores devueltos

A diferencia de C, PHP puede devolver cualquier número de valores, sólo hará falta recibir estos argumentos de la forma adecuada. Ejemplo:

```
function numeros()
  {
    return array(0,1,2);
  }
list ($cero, $uno, $dos) = numeros();
```

9.3- Funciones Variables

PHP soporta el concepto de funciones variable, esto significa que si una variable tiene unos paréntesis añadidos al final, PHP buscará una función con el mismo nombre que la evaluación de la variable, e intentará ejecutarla.

```
<?php
funtcion foo()
  {
    echo "En foo()<br>\n";
  }
function bar ($arg ='')
  {
    echo " bar();El argumento ha sido '$arg'.<br>\n";
  }
$func = 'foo';
$func();
$func='bar';
$func('test');
?>
```