**Variables:**

* Todas las variables comienzan con el símbolo del dólar ($).
* No hace falta definir una variable antes de usarla.
* Una variable no tiene un tipo asignado, esto es, podemos utilizar una variable como un entero, y después como un carácter.

Ejemplo:

<html>

<body>

<?php

$a = "Hola";

echo $a,"<br>";

$a = 1;

$b = "Ejemplo ";

echo $b,$a;

?>

</body>

</html>

*<br> introduce un salto de línea.*

*Echo imprime el valor que contienen las variables definidas previamente.*

Tenemos dos tipos de variables: locales y globales. Las variables globales tienen su ámbito de uso fuera de las funciones. Podemos acceder a ellas con la instrucción:

global nombre\_variable;

**Operadores Aritméticos:**

Operadores para aplicar a las variables y constantes numéricas.

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Nombre | Ejemplo | Descripción |
| + | Suma | 1+4 | Suma dos números |
| - | Resta | 3-2 | Resta dos números |
| \* | Multiplicación | 4\*2 | Multiplica dos números |
| / | División | 9/12 | Divide dos números |
| % | Módulo | 5%2 | Devuelve el resto de dividir dos números. |
| ++ | Suma1 | $a++ | Suma 1 al contenido de una variable. |
| -- | Resta1 | $b-- | Resta 1 al contenido de una variable. |

Ejemplo:

<html>

<body>

<?php

$a = 4;

$b = 7;

echo "$a + $b = ", $a + $b, "<br>";

echo "$a - $b = ", $a - $b, "<br>";

echo "$a \* $b = ", $a \* $b, "<br>";

echo "$a / $b = ", $a / $b, "<br>";

echo "$a % $b = ", $a % $b, "<br>";

echo "$a++ = ";

$a++;

echo $a, "<br>";

echo "$b-- = ";

$b--;

echo $b,"<br>";

?>

</body>

</html>

</body>

</html>

**Operadores de comparación:**

Operadores para comparar valores.

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Nombre | Ejemplo | Descripción |
| == | Igual | $a == $b | $a es igual $b |
| != | Distinto | $a != $b | $a es distinto a $b |
| < | Menor que | $a < $b | $a es menor que $b |
| > | Mayor que | $a > $b | $a es mayor que $b |
| <= | Menor o igual | $a <= $b | $a es menor o igual que $b |
| >= | Mayor o igual | $a >= $b | $a es mayor o igual que $b |

Ejemplo:

<html>

<body>

<?php

$a = 8;

$b = 3;

$c = 3;

echo "$a == $b = ", $a == $b,"<br>";

echo "$a != $b = ", $a != $b,"<br>";

echo "$a < $b = ", $a < $b,"<br>";

echo "$a > $b = ", $a > $b,"<br>";

echo "$a <= $c = ", $a >= $c,"<br>";

echo "$b >= $c = ", $b <= $c,"<br>";

?>

</body>

</html>

**Operadores lógicos:**

Evalúan varias comparaciones.

|  |  |  |  |
| --- | --- | --- | --- |
| Operador | Nombre | Ejemplo | Descripción |
| && , and | Y | (2>4) && (9>=2) | Cierta si ambas condiciones son ciertas |
| || , or | O | (2>4) or (9>=2) | Cierta si alguna de las condiciones es cierta |
| ! | No | ! (9>=2) | Niega la condición |

Ejemplo:

<html>

<body>

<?php

$a = 8;

$b = 3;

$c = 3;

echo "($a == $b) && ($c > $b) = ", ($a == $b) && ($c > $b),"<br>";

echo "($a == $b) || ($b == $c) = ", ($a == $b) || ($b == $c),"<br>";

echo "!($b <= $c) = ", !($b <= $c),"<br>";

?>

</body>

</html>

**Sentencias condicionales:**

|  |
| --- |
|  |

Con las sentencias condicionales podemos ejecutar unas instrucciones u otras dependiendo del resultado que obtengamos tras evaluar una condición. Las más comunes son:

* If ... else:

<?php   
   if (condición)   
   {   
 Instrucciones cuando la condición es cierta.   
   }   
   else   
   {   
      Instrucciones cuando la condición es falsa.   
   }   
?>

Ejemplo:

<html>

<body>

<?php

$a = 2;

$b = 2;

if ($a == $b) {

echo "a es igual a b";

} else {

echo "a es distinto de b";

}

?>

</body>

</html>

Como a es igual a b, se ejecuta la sentencia contenida en el ‘if’

\*Break se usa para detener el bucle y seguir a partir de este.

\*Continue sirve para ir al principio del bucle desde cualquier punto de este.

* Switch:

Con la instrucción switch podemos escoger ejecutar unas sentencias u otras dependiendo del valor de una variable.

<?php

Switch($variable){

case “valor1”:

Instrucciones si ($variable == “valor1”)

break;

case “valor2”:

Instrucciones si ($variable == “valor2”)

break;

….

default:

Instrucciones si no se da ninguno de los casos anteriores.

}

Ejemplo:

<html>

<body>

<?php

$valor = 4;

switch($valor) {

case “1”:

echo "Valor == 1";

break;

case "4": // Bloque 2

echo "Valor == 4";

break;

default: // Bloque 3

echo "La variable contiene otro valor";

}

?>

</body>

</html>

**Bucles:**

Los bucles nos permiten ejecutar una serie de instrucciones mientras se cumpla una cierta condición.

* While:

Las instrucciones se ejecutan hasta la que la condición deje de cumplirse.

<?php

while (condición){

intrucciones a ejecutar.

}

?>

Ejemplo:

<html>

<body>

<?php

$i=0;

while ($i<10)

{

echo "El valor de i es ", $i,"<br>";

$i++;

}

?>

</body>

</html>

* For:

El primer parámetro sirve para inicializar el valor de la variable que nos sirve para evaluar la condición.

El segundo parámetro indica la condición que debe cumplirse para que las instrucciones del for sigan ejecutándose.

El tercer parámetro modifica la el valor de la variable de condición, y se ejecuta al final de cada iteración.

<?php

for (inicial ; condición ; ejecutar en iteración) {

intrucciones a ejecutar.

}

?>

Ejemplo:

<html>

<body>

<?php

for($i=0 ; $i<10 ; $i++) {

echo "El valor de i es ", $i,"<br>";

}

?>

</body>

</html>

**Arrays:**

Un array es un conjunto de variables agrupadas bajo un único nombre. Cada variable dentro de la matriz se denomina elemento. Dentro de la misma matriz pueden existir variables de diferentes tipos y no es necesario que sean todas del mismo tipo.

Hay dos tipos de matrices :

* Indexada: Se accede a los elementos por la posición que ocupan en el array (la primera posición del array es la 0). Ej: $numeros[0].
* Asociativa: Los elementos tienen asociada una clave, y para acceder a ellos tenemos que acceder con el valor de esta. Ej: numeros[‘par’].

Para crear matrices existen dos formas:

* De forma implícita: Podemos introducir valores dando la posición o la clave y su valor (Ej: $numeros[‘par’] = 4), o en su defecto sin indicar ninguna clave, en cuyo caso la posición sería la siguiente a la última utilizada (Ej: $numeros[] = ’7’. Si la última posición usada fuese la 3, el 7 estaría en la 4).
* Mediante array: (‘elem1’ => ‘valor1’, ‘elem2’ => ‘valor2’,…, ‘elemN’ => ‘valorN’). Ej: $persona = array (nombre => ‘Pepe’, edad => 22);

Los arrays pueden ser multidimensionales, esto es, podemos accedes a componentes que requieran varios parámetros para identificarlas. Ej: $persona[‘Pepe’][22];

**Recorrido de una matriz:**

Para recorrer una matriz indexada, nos basta con saber el número de elementos de esta. Podemos averiguarlo mediante la función **count**(array), donde array es la matriz de la que queremos saber el número de elementos.

Para acceder a los elementos de una matriz asociativa debemos usar la función **ecah()** que recupera el par formado por clave y valor y además avanza una posición de puntero. Su sintaxis es **each(**matriz) y los valores que devuelve la matriz asociativa son:

|  |  |
| --- | --- |
| Clave | Significado |
| 0,key | Nombre de la clave |
| 1,value | Valor asociativo de la clave |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Como a la hora de recorrer las matrices asociativas no nos movemos por la posición del elemento, requerimos entre otras, las funciones siguientes:

|  |  |
| --- | --- |
| Función | Acción |
| reset (matriz) | El puntero interno va a la primera posición |
| end (matriz) | El puntero interno va a la última posición |
| next (matriz) | El puntero interno va al elemento siguiente |
| prev (matriz) | El puntero va al elemento anterior |
| current (matriz) | Devuelve el contenido del elemento actual |

Para manipular las matrices tenemos una serie de funciones que podemos encontrar en la página [http://www.webtaller.com/manual-php/matrices-array.php](http://www.webtaller.com/manual-php/matrices-array.php%20) utilizada como fuente de información para esta sección.

**Salida:**

Sentencia printf:

Podemos imprimir cadenas de texto intercalando valores que le indicamos como argumentos de la función.

<?php

printf(cadena formato, variable1, variable2...);

?>

Existen distintos formatos para formatear las variables en el printf:

|  |  |
| --- | --- |
| Operador | Tipo |
| %d | Entero |
| %f | Decimal |
| %c | Carácter ASCII |
| %s | Cadena de caracteres |

Con %.**x**f podemos decir el número de decimales que queremos que se muestren del número en cuestión (con x).

Ejemplo:

<html>

<body>

<?php

$texto="printf";

$num=1;

printf("Prueba número <b>%d</b> del %s<br>",$num,$texto);

?>

</body>

</html>

Podemos dar formato al texto, rodeándolo con <x>”texto”</x>, donde x puede tomar los siguientes valores:

|  |  |
| --- | --- |
| Operador | Tipo |
| b, strong | Negrita |
| i,em | Cursiva |
| u | Subrayado |
| s,strike,del | Tachado |
| big | Hace la letra un punto más grande(si se ponen varias, aumenta el número de etiquetas usadas) |
| small | Hace la letra un punto más grande(si se ponen varias, disminuye el número de etiquetas usadas) |
| sub | Subíndice |
| sup | Superíndice |
| blink | Parpadeo del texto |
| center | Coloca el texto en el centro de la pantalla |

Es importante indicar que si utilizamos múltiples etiquetas rodeando al texto para darle formato, el orden de apertura y cierre debe coincidir (de la más interior a la más exterior).

Además, podemos escribir las etiquetas tanto en mayúsculas como en minúsculas.

**Funciones:**

Gracias a las funciones podemos agrupar una serie de instrucciones de modo que podremos utilizarlas varias veces, desde distintos sitios sin tener que reescribirlas cada vez. Además podemos agregarles parámetros de forma que la función pueda usarlos si los necesita.

Una función puede terminar si escribimos la instrucción ‘return valor;’. En ese caso dicha función devuelve el valor retornado.

<?php

function nombre\_funcion(parametro1, parametro2,...) {

Instrucción1;

Instrucción2;

…

InstrucciónN;

return valor\_de\_retorno;

}

?>

Ejemplo:

<?php

function dame\_mayor($a, $b)

{

If ($a>=$b)

$mayor = $a;

else

$mayor = $b;

return $mayor;

}

echo dame\_mayor(4,6),"<br>";

echo dame\_mayor(3,3),"<br>";

?>

Muchas veces necesitamos usar el mismo código en distintos ficheros. Para ellos utilizaremos la función **include**(‘fichero’), que incluye el texto de ‘fichero’ en el actual ,de forma que podemos utilizar dicho código.

Ejemplo:

**libsaluda.php**

<?php

function saluda($p1){

printf("Buenos dias, %s <br> <br>",$p1);

}

function despide($p2){

printf("Buenas noches, %s <br>",$p2);

}

?>

**saludo.php**

<html>

<body>

<?php

include "libsaluda.php";

saluda("Pepe");

despide("Juan");

?>

</body>

</html>

Doc:

<http://tutorialphp.net/> (capitulo 1 2 3

<http://html.hazunaweb.com/103.php> (formato de texto

<http://www.webtaller.com/manual-php/matrices-array.php> (Matrices)

<http://php.net/manual/en/function.include.php> (Includes)