

Teoría 1

Conceptos de Programa, Proceso, Procesador, Variables, Sentencias, etc.

(Ver asimilación con recetas)

Inserción de código PHP en página HTML

```
<?php echo ' '; ?>
<html>
  <table align=center border=1 bordercolor=red>
    <tr>
      <td>
        Texto 1
      </td>
      <td>
        Texto 2
      </td>
      <td>
        Texto 3
      </td>
    </tr>
  </table>
<br/><br/><br/>
  <table align=center border=1 bordercolor=green>
    <tr>
      <td>
        <?php echo 'Fecha/hora: ' . date('Y-m-d H:i:s'); ?>
      </td>
      <td>
        <?php
          echo ' Nro aleatorio: ' . rand(1,100) . ' ';
        ?>
      </td>
      <td>
        <?php
          echo 'Texto c';
        ?>
      </td>
    </tr>
  </table>
</html>
<?php echo ' '; ?>
```

Comentarios y secuencia

```
<?php
# Este codigo imprime un mensaje fijo (este es un comentario de una linea)

/* Este es un ejemplo para probar como mostrar un mensaje, ademas para que
   salte de renglon al siguiente en HTML se puede imprimir el TAG <br>
   (este es un comentario de multiples lineas) */

    echo 'Esto es una Prueba<br>';    // Ya se imprimio el mensaje (este es un
comentario de una linea)
    echo 'segundo renglon';

?>
```

Salto de linea con PHP

```
<?php
    echo 'esto';
    echo 'sale';                // qué feo, no pone espacios en el medio, cómo
lo arreglo?
    echo 'pegado';
    echo ' mejoró :)';          // ahora sí
    echo '<br>';
    echo 'saltó !';
    echo ' y sigo en el mismo :(';
?>
```

Variable

```
<?php
// Ejemplo de variable de tipo cadena (String), asignacion y utilizacion

    $texto = 'Esto es una Prueba<br>'; // Se asigna un texto a una variable
    echo $texto; // Se imprime la variable
    print $texto . '<br>'; // Se imprime la variable y se utiliza el
operador . (concatenación de strings)
    print $texto . '<br>'; // Se imprime la variable y se utiliza el
operador . (concatenación de strings)
?>
```

Variable - Segundo Ejemplo

```
<?php
// Otro caso de concatenacion y evaluacion de variables en cadenas
$appellido = 'Rodriguez';
$nombre = 'Juan';
$apyno = $appellido . ', ' . $nombre;

echo '<pre>';
// Imprime: Rodriguez, Juan
print $appellido . ', ' . $nombre . '<br>'; // Concatenacion
// Imprime: $nombre $appellido \n
print '$nombre $appellido \n'; // Las variables y caracteres especiales
NO se interpretan dentro de '
// Imprime: Rodriguez, Juan
print $apyno . '<br>';

// Imprime: Rodriguez
print $appellido . '<br>';

$appellido .= $nombre . ' pepe';
// Imprime: RodriguezJuan pepe
print $appellido . '<br>';

echo '</pre>';
?>
```

Enteros y Reales

```
<?php
// Variables con valores enteros y reales/float/double
$cantidad = 1;
$importe = 126.50;
echo '<pre>';
print $cantidad; // Imprime: 1
print "\n";
print $importe . "\n"; // Imprime: 126.5

/* Operadores utilizados: + - (- unario) * / %
Precedencia de operadores:
Por ejemplo, en la expresión 1 + 5 * 3, la respuesta es 16 y no 18 */

print '1 + 5 * 3 => '; // Imprime: 1 + 5 * 3 =>
print 1 + 5 * 3; // Imprime: 16
print "\n";
print '(1 + 5) * 3 => ' . (1 + 5) * 3 . "\n"; // Imprime: (1 + 5) * 3 =>
18
```

```
// Otra forma de operar: += -= *= /= %=  
$cantidad += 1;  
print "La cantidad ahora es " . $cantidad . "\n"; // Imprime: La  
cantidad ahora es 2  
// Otra forma más de operar: ++ --  
print "Sumo despues y " . $cantidad++ . "\n"; // Imprime: Sumo despues y  
2  
print "Muestro " . $cantidad . "\n"; // Imprime: Muestro 3  
print "Sumo antes y " . ++$cantidad . "\n"; // Imprime: Sumo antes y 4  
echo '</pre>';  
?>
```

Booleanos

```
<?php  
echo '<pre>';  
// Variables booleanas (logicas)  
$senial1 = True;  
$senial2 = FALSE; // no importan mayusculas y minusculas en las  
constantes booleanas  
print "senial1 vale: " . $senial1 . "\n"; // Imprime: senial1 vale: 1  
$senial1 = !$senial1;  
print "senial1 vale: " . $senial1 . "\n"; // Imprime: senial1 vale:  
$senial1 = !$senial1;  
print "senial1 vale: " . $senial1 . "\n"; // Imprime: senial1 vale: 1  
// Operadores logicos: !, AND, &&, OR, || y XOR  
print "senial1 AND senial2 vale: " . ($senial1 AND $senial2) . "\n"; //  
Imprime: senial1 AND senial2 vale:  
print "senial1 && senial2 vale: " . ($senial1 && $senial2) . "\n"; //  
Imprime: senial1 && senial2 vale:  
print "senial1 OR senial2 vale: " . ($senial1 OR $senial2) . "\n"; //  
Imprime: senial1 OR senial2 vale: 1  
print "senial1 || senial2 vale: " . ($senial1 || $senial2) . "\n"; //  
Imprime: senial1 || senial2 vale: 1  
print "senial1 XOR senial2 vale: " . ($senial1 XOR $senial2) . "\n"; //  
Imprime: senial1 XOR senial2 vale: 1  
echo '</pre>';  
?>
```

La diferencia entre AND y && y entre OR y || es la precedencia.

Consultar <http://www.php.net/manual/es/language.operators.php#language.operators.precedence>.

Selección

```
<?php
```

```

    echo '<pre>';
// Ejemplo de condicionalidad (selección)
$numero = 601;
if ($numero % 2 == 0) { // % => módulo
    print "El número $numero es par\n";
    print "porque su división por dos da resto cero.";
} else
    print "El número $numero es impar";
print "\n";

if ($numero > 500)
    print "El número $numero es mayor a 500";
elseif ($numero % 2 == 0)
    print "El número $numero es par";
else
    print "El número $numero es impar";
echo '</pre>';
?>

```

Notar los siguientes puntos:

- Indentación
- El else es optativo
- Los bloques se encierran entre {} (llaves); si el bloque tiene una sola línea, el uso de llaves es opcional
- El simbolo para comparacion es == , porque = es de asignacion
- El print \n se hace fuera de la condicion

Repetición

```

<?php
    echo '<pre>';
// Ejemplo de iteracion (repeticion)
$cantidad = 10;
while ($cantidad > 1)
    print "La cantidad es " . --$cantidad . "\n";
print "El valor final de cantidad es $cantidad \n";

// En un while puede ser que no se ejecute nunca la sentencia??????
// Que pasaria si el print dentro del while fuera:
//     print "La cantidad es " . $cantidad . "\n";
// (nos olvidamos de decrementar $cantidad)??????
echo '</pre>';
?>

```

Algunos comentarios:

- Estructura de bloque para más de una sentencia mediante { }
- La condición debe ser alcanzable ! ! ! !

- La acción debe hacer que la condición esté más “cerca” de cumplirse
- Cuando terminó el ciclo es **seguro** que la condición se cumple

Repetición - Do While

```
<?php
    echo '<pre>';
// Ejemplo de repetición con DO .... WHILE ()
    $cantidad = 1;
    do {
        print "$cantidad\n";
        $cantidad++;
    } while ($cantidad < 4);
    print "Resultado final de cantidad: $cantidad";
    echo '</pre>';
?>
```

Notar que con DO WHILE la sentencia **seguro** que se ejecuta al menos una vez

Repetición - For

```
<?php
    echo '<pre>';
// Ejemplo de ciclo con FOR
    for ($indice = 1; $indice <= 5; $indice++) {
        print $indice . "\n";
    }
    echo '</pre>';
?>
```

Observaciones:

- Forma y momento de evaluación de las tres expresiones
- Usar siempre { } ?
- El ciclo **siempre se completa** ! ! !
- Luego del FOR, al igual que en el WHILE se sabe que la condición se cumple

From:
<https://wiki.rec.unicen.edu.ar/wiki/> - Wiki UNICEN

Permanent link:
<https://wiki.rec.unicen.edu.ar/wiki/doku.php?id=programacionphp3:teorias:teoria1&rev=1426370030>

Last update: 2017/10/10 16:08

