Módulo 2 **Programación en PHP y Bases de datos**

Lenguaje PHP

Objectivos y contenidos

En este módulo se verán las características más importantes del lenguaje PHP.

1. Sintaxis básica de PHP.
2. Variables: tipos y declaración. Constantes.
3. Expresiones y operadores.
4. Sentencias de control.

Sintaxis básica de PHP

El objetivo de este punto es ver las diferencias entre HTML y PHP y comenzar a familializarnos con la sintaxis de este lenguaje.

"Bienvenidos a PHP"

Escribiremos una página en HTML y otra en PHP con el mismo contenido, en PHP añadiremos comentarios, variables e incluiremos ficheros con código externo.

Bienvenidos a PHP en HTML

El código que se puede ver a continuación es muy sencillo. Es un simple mensaje de saludo en HTML.

```
<html>
Bienvenidos a <b>PHP</b><br>
<i>Saludos</i>
</html>
```

Listado 1. "Bienvenidos" en HTML

Hacemos lo mismo en PHP

El resultado será identico pero se generará de forma diferente en PHP.

Copiar el código que aparece a continuación en vuestro editor y guardarlo con el siguiente nombre *hola.php*. Observar que el resultado es el mismo.

```
<html>
HTML<br>
------------------<br>
Bienvenidos a <b>PHP</b>
<br>
<i>Saludos</i>
<br><br><br>


<?php
echo "PHP<br>";
echo "------------------<br>";
echo "Bienvenidos a <b>PHP</b>";
echo "<br>";
echo "<i>Saludos</i>";
echo "<br><br><br>";
?>
</html>
```

Listado 2. "Bienvenidos" en PHP

Si analizamos el código podremos observar las siguientes cosas:

- El código PHP debe ir precedido por el siguiente signo **"<?php"** y se señala el final del

código con **"?>"** (también se acepta iniciarlo solamente con **"<?"**). Además podremos incluir el código en cualquier sitio y tantas veces como sea necesario.

- En PHP podemos escribir código HTML, esto lo conseguiremos mediante la orden **echo**. Todo lo que este detrás de esta sentencia y tenga una correcta sintaxis se mostrará en la página resultante. PHP no analiza la validaz del código HTML que lleva insertado, este lo realizará el navegador correspondiente.

- Por último es necesario marcar el final de línea con un punto y coma **(;)**. Si no lo hacemos se producirá un error a la hora de generar la página.

Normalmente los servidores estan configurados para que revisen el código incluido en páginas que tienen la extensión .php. Por lo que, aunque nosotros escribamos código php en una página cuya extensión es .html , el servidor no lo interpretará y, por lo tanto, el resultado no será el esperado.

Saltos de línea.

En el navegador, comprobar el código HTML creado, visualizando la página anterior. Recordad que se puede ver con la opción de menú **Ver | Código fuente** (Internet Explorer 6.0) o bien **Ver | Origen de la página** (Netscape 7.0) y obtendremos algo parecido a lo siguiente.

```
<html>
HTML<br>
----------<br>
Bienvenidos a <b>PHP</b>
<br>
<i>Saludos</i>
<br><br><br>

PHP<br>----------<br>Bienvenidos a
<b>PHP</b><br><i>Saludos</i><br><br><br></html>
```

Listado 3. Código HTML generado por PHP sin saltos de línea.

Los saltos de línea escritos directamente en el editor no se correponden con los generados mediante el código PHP. Esto no afecta al funcionamiento de la página HTML, pero puede llegar a ser confuso, a la hora de analizar el código HTML creado. Podemos solucionarlo añadiendo un salto de línea de código (**\n**) al final de cada línea. Esta línea no afecta el código HTML generado pero si que producirá una presentación más clara cuando veamos el código fuente.

```
<?php
echo "PHP<br> \n";
echo "----------<br> \n";
echo "Bienvenidos a <b>PHP</b> \n";
echo "<br> \n";
echo "<i>Saludos</i> \n";
echo "<br><br><br> \n";
?>
```

Listado 4. Saltos de línea.

Y el resultado esta vez sería:

```
<html>
HTML<br>
----------<br>
Bienvenidos a <b>PHP</b>
<br>
<i>Saludos</i>
<br><br><br>
```

```
PHP<br>
------------------<br>
Bienvenidos a <b>PHP</b>
<br>
<i>Saludos</i>
<br><br><br>
</html>
```

Listado 5. Código HTML generado en PHP con saltos de línea

Signos de escape.

Para hacer un salto de línea en el código, tenemos que utilizar un signo de escape (\) en combinación con otro carácter (\n).

La siguiente tabla muestra algunos de los signos de escape más utilizados:

código	resultado
\n	Crea una nueva línea.
\r	Crea un salto de parrafo.
\t	Crea una tabulación.

Igualmente, este signo puede servir para "emmascarar" signos especiales, que se utilizan ya en la sintaxis del lenguaje, como:

código	resultado
\'	Comillas simples.
\$	Signo dólar (este signo precede a todas las variables en PHP).
****	La misma barra invertida.
\"	Comillas dobles.

En el código siguiente la palabra PHP de la última frase será mostrada entre comillas dobles, gracias al enmascaramiento. En caso contrario, quedaría cortada la cadena y daría un error. También se podría evitar este problema alternando comillas simples.

```
<?php
echo "PHP<br> \n";
echo "------------------<br> \n";
echo "Bienvenidos a <b>PHP</b> \n";
echo "<br> \n";
echo "<i>Saludos</i> de \"PHP\" <br> \n";
echo "<br><br><br> \n";
?>
```

Listado 6. Emmascaramiento mediante signos de escape.

Comentarios en el código.

Es recomendable comentar el código que escribimos, esto nos ayudará:

- A recordar qué hace cada fragmento de código.
- A colocar observaciones o advertencias.
- A comentar, en el proceso de depuración, una línea para ver el funcionamiento y encontrar errores.

```
<html>
<?php

/*
Práctica 1
Módulo 2
Curso de PHP
(esto es un comentario de más de una línea)
*/

echo "PHP<br> \n";
echo "------------------<br> \n";
echo "Bienvenidos a <b>PHP</b> \n";
echo "<br> \n";

// Ahora explicaremos la inclusión de comillas dobles en una cadena
// \n sirve para hacer saltos de línea en el código HTML generado
echo "<i>Saludos</i> de \"PHP\" <br> \n";
echo "<br><br><br> \n";
?>
</html>
```

Listado 7. Código comentado.

Como se puede ver podemos hacer comentarios de más de una línea colocandolos entre **/*** y ***/.**

Igualmente, podemos colocar un comentario de una sola línea después del signo **//**.

Variables: tipos y declaración.
Inclusión de ficheros.

El objetivo de este punto es hacer una introducción a las posibilidades de PHP para trabajar nuevas características de la sintaxis y realizar la inserción de ficheros de código. Se verá:

- Variables (tipos y declaración).
- Constantes.
- Inclusión de ficheros externos y bibliotecas.

Variables.

En PHP podemos encontrar 5 tipos de variables: **Integer, Double, String, Array y Object.**

PHP gestiona de manera automática el tipo de variable y hace las conversiones convenientes.

```
<?php
/*-------------------------------------------------------------------
* Módulo: 2 Práctica: 2
* Autor:  Data:
* Descripción: Tipos de variables en PHP
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/ // definición de variables
$nombre = "Andrea"; // variables de texto o cadena (String)
$apell1 = "León";
$apell2 = "Garcia";
$edad = 37; // variable numérica (Integer)
$pobla = "Valencia"; //Inserción de variables dentro de la cadena
echo "Tu nombre és: $nombre <br> \n";
echo "Tus apellidos són: $apell1 y $apell2 <br> \n";
echo "Tienes $edad años <br> \n";
echo "Vives en $pobla <br> \n";
?>
```

Listado 1. Definición y presentación de variables

Aclararemos algunos aspectos básicos:

- Los nombres de las variables deben de comenzar con el signo dólar (**$**) seguido de una cadena de caracteres que comienza por un letra o un subrayado y que sigue con cualquier número de letras, números o subrayados.

- El uso de mayúsculas y minúsculas esta permitido, pero hay que tener en cuenta que el interprete las diferencia. El mismo nombre con al menos una letra en mayúscula o minúscula diferente identifica a variables distintas.

- No es necesario declarar el tipo de variable. PHP reconoce automáticamente el tipo de variable cuando se le hace una asignación.

- Podemos elegir el nombre que queramos para las variables, pero es recomendable que tengan un significado descriptivo de para qué va a servir.

- No podemos utilizar como nombre de variable ninguna palabra reservada del lenguaje PHP.

- No es recomendable poner acentos o caracteres especiales en los nombres de variable (ç,ñ...), aunque podría funcionar correctamente.

Si vemos el listado 1 de este punto, veremos que se definen cuatro variables de tipo cadena (**$nombre**, **$apell1 $apell2** y **$pobla**, los valores asignados están entre comillas y una variable numérica (**$edad**), con el valor sin comillas.

Después de la definición, se muestran los valores utilizando la sentencia **echo**. Hay que fijarse en que utilizamos dobles comillas (**"**) para el contenido de **echo,** y podemos insertar directamente las variables dentro de la cadena, da igual el tipo de variable de que se trate:

```
echo "Tienes $edad años <br> \n";
echo "Vives en $pobla <br> \n";
```

A veces, este sistema de inserción puede ocasionar algún problema de interpretación, por lo que en el siguiente módulo, donde se verá la concatenación de cadenas y variables, veremos un sistema más seguro.


La inserción directa de variables en la cadena no funciona si esta esta entre comillas simples ('): **echo 'Vives en $pobla
 \n';** no funcionará.


PHP aporta algunas funciones muy interesantes para trabajar con variables, entre las cuales podemos destacar:

función	**resultado**
gettype(variable)	Devuelve el tipo de la variable pasada como argumento: *integer, float, string, array, class, object* o *unknown type.*
settype(variable, tipo)	Recibiendo como parmetros una referencia a una variable y una cadena de caracteres, cambia el tipo de la variable referenciada al indicado por dicha cadena.
isset(variable)	Devuelve TRUE si la variable pasada está ya definida. False en otro caso.

Constantes.

PHP permite la definición de constantes, el valor asignado permanecerá invariante a lo largo de la ejecución del programa, para hacerlo se utiliza la siguiente función:

define("constante",valor);

Al hacer referencia a una constante no es necesario poner el signo dólar delante del nombre de la constante, puesto que en tal caso haría referencia a una variable.

Inclusión de código desde un fichero.

Veremos ahora como incluir un fichero que nos servirá como encabezado de la página:


Copiar el siguiente código en vuestro editor y guardarlo como practica22.*php*

```
<?php
/*-------------------------------------------------------------
* Módulo: 2 Práctica: 2
* Autor:
* Descripción:
```

```
* Pre condi.:
* Post cond.:
-----------------------------------------------------------------*/
// Definición de constantes, normalmente en mayúsculas
// definimos la constante del título del curso
define("TITCUR","Programación en PHP y Bases de datos");
// definimos la constante para el salto de línea HTML
define("B","<br>");

// inclusión de un fichero externo
include( "cabecera.php" );

echo B;
echo B;
echo B;
echo "PHP<br> \n";
echo "------------------<br> \n";
echo "Bienvenidos a <b>PHP</b> \n";
echo "<br> \n";

echo "<i>Saludos</i> de \"PHP\" <br> \n";
echo "<br><br><br> \n";
?>
```

Listado 2. Código del fichero principal

Copiar también el código que ha continuación aparece y guardarlo como cabecera.*php*

```
<table style="text-align: left; width: 633px;" border="1"
 cellpadding="2" cellspacing="2">
  <tbody onload="loadPage()">
    <tr>
    <td>
<?php echo "Programación en PHP y Bases de datos" ?><br>
<?php echo "Fecha: ".date("d/m/Y"); ?> </tr>
  </td>
  </tbody onload="loadPage()">
</table>
```

Listado 3. Código del fichero incluido: 'cabecera.php'

En el código de practica22.php se hacen 2 cosas distintas:

Por una parte, se han definido unas constantes utilizando la función **define()**.

define("TITCUR","Programación en PHP y Bases de datos");

Cada vez que utilicemos TITCUR en cualquier sitio del código, será sustituido por el valor que le hemos asignado.

**define("B","
");**

También se ha definido la constante *B*, que contendrá la etiqueta predefinida
 del código HTML.

Por otro lado, se ha incluido la sentencia **include()** para incluir fichero externo.

include("cabecera.php");

El código del fichero cabecera.php será incluido en el lugar en que se colocó la sentecia *include*

y funcionará como si fuera un solo fichero.

La función **include_once()**no permite la inclusión del fichero más que una vez como máximo en un script. Igualmente, también existe **require()** y **require_once()**. La diferencia de *include* y *require* es que este último no permite la inclusión condicional, es decir, que aunque la sentencia este situada dentro de una sentencia alternativa siempre se incluirá el codigo, pase o no la ejecución del programa por ese punto.

En el fichero cabecera.php, también se utiliza otra función de PHP: **date()**. Dependiendo de los parametros que se le pasen, nos devolverá la fecha del sistema en el momento de ejecución del código en formatos diferentes.

Hay que tener en cuenta la ubicación del fichero cabecera.php. En el ejemplo que hemos visto suponemos que se encuantra en el mismo directorio que el fichero principal. En caso contrario, se debería indicar el camino donde se encuentra el fichero.

Si se incluye un fichero en el que se encuentra código php, se debe indicar la apertura y el cierre de este código con **<?php** y ***?>.***

Expresiones y Operadores

El objetivo de este módulo es familiarizarse con los diferentes tipos de operadores que nos ofrece PHP y como construir expresiones validas que nos permitirán:

- hacer cálculos.
- realizar asignaciones.
- establecer comparaciones lógicas.

Operadores

Los operadores que veremos serán los siguientes:

- aritméticos.
- de asignación.
- comparación.
- ejecución.
- incremento y decremento.
- lógicos.

Operadores aritméticos

Permiten hacer cálculos básicos y son los siguientes:

ejemplo	nombre	resultado
$a + $b	Suma	Suma de $a y $b
$a - $b	Resta	Resta entre $a y $b
$a * $b	Multiplicación	Producto de $a y $b
$a / $b	División	Cociente de $a entre $b
$a % $b	Mòdulo	Resto de $a dividido entre $b

Por ejemplo:

```
<?php
$a = 4;
$b = 7;
$c = $a + $b; //la variable $c será igual a 11

echo "\$a=".$a."<br> \n";
echo "\$b=".$b."<br> \n";
echo "\$c=\$a+\$b <br><br> \n";
echo "Resultado: ". $c;
?>
```

Observad que, para poder ver en pantalla el nombre de la variable y no su contenido, debemos recurrir a signo de escape **(\)** delante del signo de inici de variable **($).**

Operadores de asignación

En el ejemplo anterior hemos visto que con PHP las asignaciones básicas se hacen mediante el operador '='. Este operador hace que el operando de la izquierda coja el valor de la derecha.

Por ejemplo: $total = 2500;

la variable $total tendrá el valor 2500

Pero también hay operadores combinados, que permiten asignar y operar en un sólo paso:

```
$a = 4;
$a += 6;     //  esto es equivalente a  $a = $a + 6
```

y también con cadenas

```
$cadena = "Hola ";
$cadena .= "José"; //  $cadena tendrá el siquiente valor "Hola José"
```

Operadores de comparación

Como en la mayoria de lenguajes, PHP tiene sus comparadores que nos permitirán tomar decisiones. La siguiente tabla muestra una relación de operadores:

exemple	nom	resultat
$a == $b	Igual	TRUE si $a es igual a $b
$a === $b	Identidad	TRUE si $a es igual a $b i si son del mismo tipo
$a != $b	Distinto	TRUE si $a no es igual a $b
$a < $b	Menor que	TRUE si $a es estrictamente menor que $b
$a > $b	Mayor que	TRUE si $a es estrictamente mayor que $b
$a <= $b	Menor o igual que	TRUE si $a es menor o igual que $b
$a >= $b	Mayor o igual que	TRUE si $a es mayor o igual que $b


Atención, el operador que valora la igualdad es un doble igual (**==**).

Operadores de incremento/decremento

Permiten incrementar/decrementar el valor de una variable en una unidad.
El incremento/decremento puede hacerse antes o después de tomar el valor de la variable

ejemplo	nombre	efecto
++$a	Preincremento	Incrementa $a en uno y después devuelve $a
$a++	Postincremento	Devuelve $a y después incrementa $a en uno
--$a	Predecremento	Decrementa $a en uno y después devuelve $a
$a--	Postdecremento	Devuelve $a y después decrementa $a en uno

Operadores lógicos

ejemplo	nombre	efecto
$a and $b	Y	TRUE si tanto $a como $b son ciertos
$a or $b	O	TRUE si $a o $b son ciertos
$a xor $b	O exclusiva	TRUE si $a o $b son ciertos, pero no los dos a vez
!$a	Negación	TRUE si $a no es cierto
$a && $b	Y	TRUE si tanto $a como $b son ciertos
$a || $b	O	TRUE si $a o $b son ciertos

Atención, en realidad sólo disponemos de cuatro operadores lógicos. **And** y **&&** son equivalentes, como **or** y **||** .

Vamos a ver un ejemplo

Aunque en el siguiente módulo veremos las estructuras de control, para poder comprobar el funcionamiento de estos operadores, avanzaremos el uso de la condición (**if** y **else**). Copia el código a tu editor y guárdalo con el nombre operador.php.

```
<?php
/*-------------------------------------------------------------------
* Módulo: 2 Práctica: 3
* Autor:  Fecha:
* Descripción: ejemplo de operadores
* Pre condi.:
* Post condi.:
-------------------------------------------------------------------*/

// Comprobación de una nota

$nota = 5; // asignaciones
$mensaje = "";

/* Lo primero es comprobar que la nota este comprendida entre 1 y 10 */


if( $nota >= 1 && $nota <= 10 )
  {
  if($nota < 5 )
    {                //Si la nota es inferior a 5
      $color="#AA0000"; // color del mensaje
      $mensaje = "No has superado la prueba.";
    }
      else{            // en caso contrario
      $color="#0000AA"; // color del mensaje
      $mensaje = "Prueba superada";
    }
  }
  else{
      $color="#000000";  // color del mensaje
      $mensaje = "Error! la nota debe estar entre 1 y 10";
  }
  echo "<font color=$color> $mensaje Calificación: $nota </font> <br>";
?>
```

Listado 2.3.1. Ejemplo con operadores.

Realizar cálculos en PHP

Para comprobar en la práctica la capacidad de PHP, veremos un sencillo ejemplo de cálculo del IVA. Copiar este código y guardalo con el nombre m2p32.php.

```
<?php
/*-------------------------------------------------------------------
* Módulo: 2 Práctica: 3
* Autor:  Data:
* Descripció: Calculo del IVA
* Pre condi.:
* Post condi.:
-------------------------------------------------------------------*/

$neto = 200; //neto
$iva  = 0.16;
$total_iva = $neto*$iva; // total iva a pagar
$total = $neto+$total_iva;

echo "<TABLE BGCOLOR=\"#AABBCC\"> \n";
echo "<TR><TD width=\"100\">Neto: </TD>
<TD>".$neto." € </TD></TR><br> \n";
echo "<TR><TD width=\"100\">IVA: (".$iva."): </TD> <TD> ".$total_iva." € </TD></TR><br> \n";
echo "<TR><TD width=\"100\">Total: </TD> <TD>".$total." €</TD></TR> \n";
echo "</TABLE> \n";
?>
```

Listado 2.3.2. Cálculo de IVA

En el ejemplo podemos ver los diferentes operadores de cálculo.

Tenemos dos variables (**$neto** y **$iva**) a partir de las ellas hacemos los cálculos pertinentes.

En la presentación de los resultados se genera una tabla en html.

Sentencias de control de programa

En este tema veremos:

- Sentencias de selección.
- Sentencias de iteración.

Las principales sentencias de control en PHP son:

sentencias	sintaxis

if, **else**, **elseif**	`if (expr)`
	`  sentencia`
	
	o bien
	
	`if (expr){`
	`  sentencias`
	`  }`
	
	`if (expr){`
	`  sentencias;`
	`  }`
	`elseif {`
	`  sentencias;`
	`elseif {`
	`  sentencias;`
	`else {`
	`  sentencias;`
	`}`

codicional compacta	`$variable = (exp1) ? exp2 : exp3;`

switch	`switch ($i) {`
	`  case 0:`
	`  echo"i es igual a 0";`
	`  break;`
	`case 1:`
	`  echo "i es igual a 1";`
	`  break;`
	`case 2:`
	`  echo "i es igual a 2";`
	`  break;`
	`default:`
	`  echo "Otro valor";`
	`}`

for	`for (expr1; expr2; expr3){`
	`  sentencia`
	`}`

	`foreach(expresion_array as $value) {` `  sentencia}`

foreach	*o bien* `foreach(expresion_array as $key => $value) { sentencia}`

while	`while (expr) {`
	`  sentencia`
	`}`

Sentencias de selección.

Estructura if, else, elseif

La estructura **if**, **else**, **elseif** es, la estructura condicional clásica, implementada en todos los lenguajes.

```
<?php
/*----------------------------------------------------------------
* Módulo: 2 Práctica: 4
* Autor:
* Descripción: Estructuras if, else, elseif
* Pre condi.:
* Post condi.:
----------------------------------------------------------------*/

$grupo = "B";

if ($grupo == "A"){
   echo "El grupo es A";
   }
   elseif ($grupo == "B")
     {
       echo "El grupo es B";
     }
   elseif ($grupo == "C")
     {
     echo "El grupo es C";
     }
   else {
      echo "El grupo no es ni A ni B ni C";
}
?>
```

Listado 2.4.1. Ejemplo estructura condicional

Condicional compacta

La sintaxis

```
$variable = (exp1) ? exp2 : exp3;
```

primero se evalua la expresión (exp1), si es cierta se asigna el resultado de evaluar la expresión (exp2) a la variable, en caso contrario se asigna a la variable el resultado de evaluar la expresión 3. Por ejemplo:

```
<?php
```


```
/*----------------------------------------------------------------
* Módulo: 2 Práctica: 4
* Autor:
* Descripción: Condicional compacta. Máximo de tres valores.
* Pre condi.:
* Post cond.:
----------------------------------------------------------------*/

$a = 5;
$b = 8;
$c = 3;

// Si $a < $b $maximo = $b sino igual a $a
$maximo = ($a<$b)? $b : $a;

// ya tenemos el mayor de $a y $b

$maximo = ($maximo<$c)? $c : $maximo;

// ya tenemos el mayor del anterior y el último número

// Escribimos el resultado

echo "El máximo de los tres valores es: $maximo";
?>
```

Listado 2.4.2. Ejemplo de condicional campacta.

Estructura switch

La estructura **switch** es equivalente a una serie de if. Cuando se cumple el caso se sale del proceso utilizando la sentecia break colocada destrás de cada evaluación.


```
<?php
/*----------------------------------------------------------------
* Módulo: 2 Práctica: 4
* Autor:
* Descripción: Ejemplo de uso de 'switch'
* Pre condi.:
* Post cond.:
----------------------------------------------------------------*/

$pla = 'ESO';

switch ($pla)
{
  case 'ESO':
      echo 'Educación Secundaria Obligatoria';
      break;
  case 'BACH':
      echo 'Bachillerato';
      break;
  case 'CFGM':
      echo 'Ciclo Formativo de Grado Medio';
      break;
  case 'CFGS':
      echo 'Ciclo Formativo de Grado Superior';
      break;
  // si no ha coincidido con ninguna de las anteriores
  default:
      echo 'Codificación incorrecta';
}
```

```
?>
```

Listado 2.4.3. Ejemplo de uso de switch.

Otro ejemplo de switch sería la elección individual de un color:

```
<html>
<head>
<title>Modificar el color de fondo</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" type="text/css" href="../css/nuevo.css">
</head>
<body bgcolor="<?php
if(isset($_POST["color"])) {
switch(($_POST["color"])) {
case 1:
echo "#ffcc33";
break;
case 2:
echo "ffff99";
break;
default:
echo "silver";
}
}
else {
echo "white";
}
?>">

<h1>Seleccione un color de fondo</h1>

<form action="color.php" method="post">
<input type="radio" name="color" value="1"> naranja<br>
<input type="radio" name="color" value="2"> amarillo<br>
<input type="radio" name="color" value="x"> sorpresa<br><br>
<input type="submit" value="Seleccionar un color">
</form>
</body>
</html>
```

Listado 2.4.4. Ejemplo de uso de switch.

Sentecias de iteración (bucles)

Podemos decir, que un bucle es una repetición de una acción o conjunto de acciones hasta que se cumpla una condición o situación final que se ha definido en la cabecera del bucle.

Hay que tener en cuenta que con la utilización de los bucles, tenemos que tener mucho cuidado en su definición, puesto que como hemos dicho en el tema anterior PHP es un lenguaje interpretado que se ejecuta en la parte del servidor, un bucle infinito podría llegar a colgar el servidor web.

For

La estructura **for** es una de las complejas y al vez flexibles de PHP.

Sintaxis:

```
for (expr1; expr2; expr3){
    sentencia
}
```

Donde:
`expr1`: inicializa la variable que usaremos en el bucle **for**.
`expr2`: define la condición que debe cumplirse para poder entrar en el proceso **for**; en el momento en que no se cumpla se acaba el proceso.
`expr3`: modifica el valor de la variable usada como contador del bucle, puede incrementar, decrementar, etc.


```
<?php
/*-------------------------------------------------------------------
* Módulo: 2 Práctica: 4
* Autor:
* Descripción: Ejemplo de 'for'
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/

$num = 10; //definimos una variable

// expr1: $i valdrá 0
// expr2: la condición ($i ha de ser diferente a $num)
// expr3: en cada pasada se incrementa $i en 1 ($i++)
for($i=0; $i != $num ; $i++)
{
echo "\$num es diferente de ".$i."<br>";
}
echo "<br><br>";
echo "Estamos fuera del bucle for <br>";
echo "\$num ($num) es igual a \$i ($i)";
?>
```

Listado 2.4.5. Ejemplo de uso de for.

Otro ejemplo de bucle for, se pregunta cuantas veces quiere que repita una frase y las escribe:

```
<html>
<head>
<title>El bucle for</title>
<meta http-equiv="content-type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" type="text/css" href="../css/nuevo.css">
</head>
<body>
<h1>El bucle for</h1>

<form action="<?php echo $_SERVER["PHP_SELF"]; ?>" method="post">
¿Cuántas veces debe aparecer la frase?
<input type="text" name="cantidad">
<input type="submit" value="enviar">
</form>
<p>
<?php
if (isset($_post["cantidad"])) {
for ($contador=1;$contador<=$_POST["cantidad"];$contador++) {
echo "<b>$contador</b>: Loops are easy!<br>\n";
}
echo "Se acabó.\n";
}
?>
</p>
</body>
</html>
```

Listado 2.4.6. Ejemplo de uso de for.

Foreach

La construcción **foreach** es la más sencilla cuando el bucle esté relacionado con un array, aunque los arrays los veremos en temas posteriores, no resultará complicado entender la estructura foreach. Una de sus dos sintaxis es::

```
foreach(expresion_array as $value) {
   sentencia}
```

Le pasamos un array como primer parametro y una variable como segundo, a cada iteracción del bucle la variable irá cogiendo un valor del array, avanzando al siguiente elemento en cada pasada, hasta llegar al final del array. Veamos un ejemplo que visualiza los días de la semana.

```
<?php
/*-------------------------------------------------------------------
* Módulo: 2 Práctica: 4
* Autor:
* Descripción: Ejemplo de uso de 'foreach'
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/

// Definimos un array con los dias de la semana
$dia[0] = "Lunes";
$dia[1] = "Martes";
$dia[2] = "Miércoles";
$dia[3] = "Jueves";
$dia[4] = "Viernes";
$dia[5] = "Sabado";
$dia[6] = "Domingo";

// Y ahora se muestran por pantalla
echo "Los días de la semana son:<br><br>\n";


foreach ($dia as $valor) {
       //en cada pasada se lee un nuevo valor del array
       echo "$valor<br>\n";
}
?>
```

Listado 2.4.7.Ejemplo de utilización de foreach.

Algunos ejemplos más para demostrar su uso:

```
<?php

/* foreach ejemplo 1: sólo valor*/
$a = array(1, 2, 3, 17);

foreach($a as $v) {
   print "Valor actual de \$a: $v.\n";
}

/* foreach ejemplo 2: valor (con clave impresa para ilustrar) */
$a = array(1, 2, 3, 17);

$i = 0; /* sólo para propósitos demostrativos */

foreach($a as $v) {
   print "\$a[$i] => $v.\n";
   $i++;
}
```

```
/* foreach ejemplo 3: clave y valor */
$a = array(
   "uno" => 1,
   "dos" => 2,
   "tres" => 3,
   "diecisiete" => 17
);

foreach($a as $k => $v) {
   print "\$a[$k] => $v.\n";
}

/* foreach ejemplo 4: matriz multi-dimensional */

$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";

foreach($a as $v1) {
   foreach ($v1 as $v2) {
      print "$v2\n";
   }
}

/* foreach ejemplo 5: matriz dinámica */

foreach(array(1, 2, 3, 4, 5) as $v) {
   print "$v\n";
}
?>
```

Listado 2.4.8.Ejemplo de uso de foreach.

While

Un alternativa al for es utilizar **while**. Se trata de una estructura parecida a *for* pero que no incluye en su declaración ni la inicialización de una variable de control (la inicialización se debe hacer antes de entrar en el bucle), ni su incremento o decremento, esta modificación de la variable de control se debe realizar en el cuerpo de la estructura. Veamos un ejemplo:

```
<?php
/*-------------------------------------------------------------------
* Módulo: 2 Práctica: 4
* Autor:  fecha:
* Descripción: Ejemplo de uso de 'while'
* Pre condi.:
* Post cond.:
-------------------------------------------------------------------*/

// Números divisibles por $div hasta $total

$i = 0;       //variable de control del while
$div = 5;     //número a dividir (divisor)
$total = 100; //total

echo "<h2>Números divisibles por $div hasta $total.</h2>";

while ($i < $total){
   // Si es divisible entre $div, lo mostramos
   if ($i%$div == 0 ){
                     echo $i." - ";
```

```
                }
  // Incremento de $i
  $i++;
  } // fin del while
?>
```

Listado 2.4.9.Ejemplo de uso de while.

Do .. While.

Los bucles do..while son muy similares a los bucles while, excepto que las condiciones se comprueban al final de cada iteración en vez de al principio. La principal diferencia frente a los bucles regulares while es que se garantiza la ejecución de la primera iteración de un bucle do..while (la condición se comprueba sólo al final de la iteración), mientras que puede no ser necesariamente ejecutada con un bucle while regular (la condición se comprueba al principio de cada iteración, si esta se evalúa como FALSE desde el principio la ejecución del bucle finalizará inmediatamente).

Hay una sola sintaxis para los bucles do..while:

```
<?php
 $i = 0;
 do {
    print $i;
 } while ($i>0);
?>
```

Listado 2.4.10.Ejemplo de uso de do .. while.

El bucle de arriba se ejecutaría exactamente una sola vez, después de la primera iteración, cuando la condición se comprueba, se evalúa como FALSE ($i no es más grande que 0) y la ejecución del bucle finaliza.

```
<?php
do {
    if ($i < 5) {
        print "i no es lo suficientemente grande";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i es correcto";
    /* procesa i */
 } while(0);
?>
```

Listado 2.4.11.Ejemplo de uso de do .. while.

Ejercicio 1. Números primos.

El ejercicio consiste en dado un número positivo (se almacenará en una variable) mostrar en una tabla todos los números primos menores que él.

Almacenar el fichero con el nombre **mod2ejer1.php.**

Ejercicio 2. Números divisibles.

El ejercicio consiste en generar una tabla que contenga verticalmente los números del 1 al 10, que serán los divisores. Horizontalmente, tendremos 10 números más (por ejemplo del 50 al 60) que serán, los dividendos. Los valores de los dividendos se deben almacenar en dos variables.

Se trata de rellenar la tabla con un asterisco (*), en la intersección de los números en el caso de ser divisible, en caso contrario se rellenará con un guión (-).

El color de fondo de cada casilla variará dependiendo del valor que contengan.

Almacenar el fichero con el nombre **mod2ejer2.php**.

Comprimir estos ficheros con el nombre **inicial-nombre_primer-apellido_mod3ej1.zip** y envíalo.