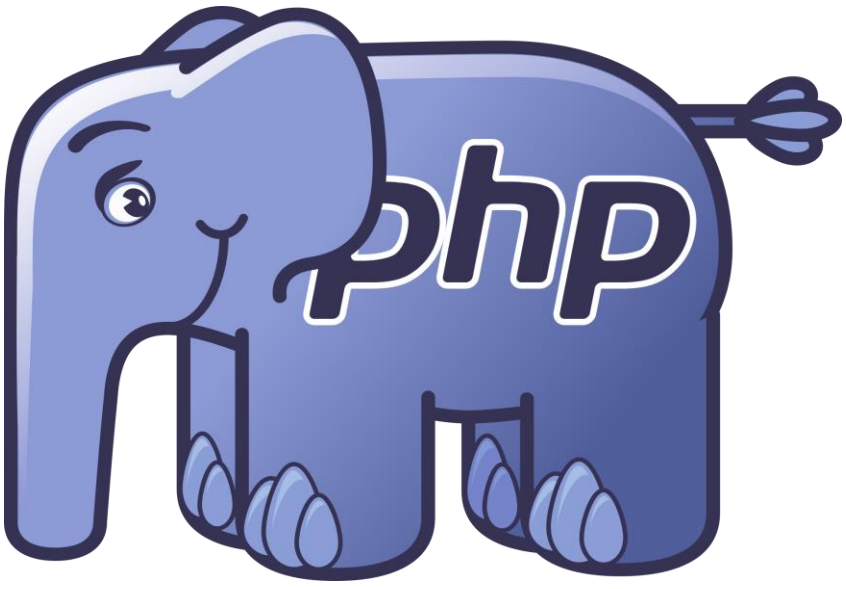
TEMA

2

2ºDAW

Empezando con PHP



1 Introducción PHP

PHP (Personal Home Page) es, según el índice PYPL[1] (PopularitY of Programming Language index), el segundo lenguaje de programación más utilizado en el mundo, únicamente superado por Java[2].

PHP es un lenguaje de programación estructurado y, como tal, hace uso de variables, sentencias condicionales, bucles, funciones… PHP es también un lenguaje de programación orientado a objetos y, por consiguiente, permite definir clases con sus métodos correspondientes y crear instancias de esas clases.

A diferencia de JavaScript o HTML que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tiene el servidor, por ejemplo a una base de datos. Los programas escritos en PHP se ejecutan en el servidor y el resultado se envía al navegador. El resultado es normalmente una página HTML.

Al ser PHP un **lenguaje dinámico de código abierto que se ejecuta en el servidor** no es necesario que el navegador lo soporte, es independiente del navegador, pero sin embargo para que las páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP. Hoy en día la práctica totalidad de servidores que ofrecen servicios de hosting soportan PHP por defecto.

¿Por qué usar PHP?	Alternativas
✓ Es fácil ✓ Amplia documentación ✓ Es de los más usado y soportado ✓ Capacidad de expansión (librerías ,framework, CMS)	✓ Python ✓ Ruby ✓ Java

Mi primer programa (con comentarios)

¿Para indicarle al servidor que escribimos código php utilizaremos la etiqueta <?**php** …… **?>.**

```
<!DOCTYPE html>
<html lang="es">
<head>
        <meta charset="utf-8">
        <title>primer php</title>
 </head>
<body>
<!-- Cuerpo de html -->
<h1>Esto es código HTML<?php echo ' y código php embebido'?></h1>
<?php
// Esto es un comentario aquí comienza el codigo php. El echo escribe por pantalla
echo "<h2>Esto es <b>PHP:</b> Hola mundo PHP</h2>";
/* Aquí tienen una lista de opciones
Concatenadas con un punto */
echo "<ul>"
   ."<li>opcion 1</li>"
   ."<li>opcion 2</li>"
   ."</ul>";
?>
```

```
<?= "<p>Esto es otra manera rápida de escribir en PHP</p>" ?>
</body></html>
```

Nota: Es más eficiente utiliza comilla simple a comilla doble

Se puede alternar entre PHP y HTML cuantas veces se quiera, sólo hay que asegurarse de incluir todas las etiquetas de apertura y de cierre.

2 Variable y constantes

Una constante es un contenedor de información que no puede variar. No comienzan con el símbolo dólar como las variables.

Una variable es un contenedor de información, es algo así como una cajita que tiene un nombre y en la que podemos meter un valor. Las variables pueden almacenar

- Enteros (Integer)=99
- Coma flotante/ decimales (float) = 3.45
- cadenas de caracteres (string)= " Hola soy un string"
- Booleanos (Bool)= true o flase
- Null.
- Array (colección de datos)
- Objetos

El contenido de las variables se puede mostrar y se puede cambiar durante la ejecución de una página PHP (por eso se llaman variables).

Los nombres de las variables comienzan con el símbolo del dólar ($) y **no es necesario definirlas (PHP es débilmente tipado)** como se hace en otros lenguajes de programación como C, Java, Pascal, etc. La misma variable puede contener un número y luego el nombre de una ciudad, no existe restricción en cuanto al tipo como en la mayoría de los lenguajes.

Mientras depuramos un programa, con frecuencia necesitamos ver el valor de las variables (con funciones sin utilizar echo).

var_dump ($variable): muestra el valor y el tipo de a variable

```
<!DOCTYPE  html>
<html>
<head>
<meta  charset="UTF-8">
</head>
<body>
<?php
// constante
define ('pi',3.1416);

//variables
$x  =  24;
$animal =  "conejo";
$saludo =  "hola caracola";
echo  'El doble de 24 es'. $x*2 . '<br/>'. pi.  '<br/>'.$animal.  '<br>'.  $saludo;

var_dump($x);
?>
</body>
</html>
```

Ejemplo2.php

```
<!DOCTYPE html>
<html lang=”es”>
<head>
        <meta charset="utf-8">
         <title>ejemplo 2</title>
 </head>
<body>
<?php
//Definimos dos variables
        $servidor=“localhost”;
        $usuario=“root”;
?>
<h1>variables</h1>
<?php
//Escribimos en pantalla las variables
        echo “<span style=’color:blue’>Nombre del servidor:</span> $servidor<br/>”;
        echo “ <span style=’color:blue’>Nombre de usuario: </span>$usuario<br/>”;
?>
</body>
</html>
```

Nota: Cuidado con el uso de comillas dobles y simples para incluir la etiqueta style de css

Ejercicios

- ✓ **Ejercicio 1:** Escribe un programa que muestre tus datos personales por pantalla (nombre, apellidos, dirección, edad , nº teléfono, dirección). Cada dato tendrá que estar almacenado en una variable. Utiliza la etiqueta Style para darle estilo.

  Nombre: Antonio
  Apellidos: ……..

- ✓ **Ejercicio 2:** Escribe un programa que utilice las variables $x y $y. Asígnales los valores 144 y 999 respectivamente. A continuación, muestra por pantalla el valor de cada variable, la suma, la resta, la división y la multiplicación.
- ✓ **Ejercicio 3:** Realiza un conversor de pesetas a euros. La cantidad en euros que se quiere convertir deberá estar almacenada en una variable.
- ✓ **Ejercicio 4:** Realizar el área y la longitud de la circunferencia utilizando la variable $radio=12. Utiliza la constante pi.

3 Paso de datos a través de URL o formularios

Metodo Get

Cuando enviamos datos con el método GET, estos son enviados en el URL y cuando la página que solicitamos termine de cargar estos datos serán visibles para el usuario. Veamos un ejemplo de esto:

```
<a href="http://localhost/pagina.php?categoria=ropa&producto=36">ver</a>
```

Observemos el URL, estamos llamando al PHP pagina.php y luego tenemos una serie de datos. Estos son los datos que estamos enviando al servidor por medio del método GET, esto se hace mediante un esquema nombre/valor. Todo lo que vaya después del signo de interrogación ? serán datos que se están enviando al servidor y para enviar varios los separamos con un signo &.Con la variable global $_GET se obtiene los datos

Pagina.php

```
$categoria = $_GET['categoria'];
$producto = $_GET['producto'];
echo "Selecciono el producto  $producto  de la categoría $categoria";
//otra forma de mostrar por pantalla la información sin utilizar las variables declaradas
echo "Selecciono el producto".  $_GET['producto'] ."de la categoría ". $_GET['categoria'];

// Estos nos imprimirá en el navegador -> Selecciono el producto 3 de la categoría ropa
// Tambien podriamos a ver puesto la variables directamente concatenando
```

También podemos enviar información por el método GET a través de un formulario HTML.

```
<form name="form" id="form" action="pagina.php" method="GET" >
  <label for="categoria">Categoria:</label>
<input type="text" name="categoria" id="categoria"/>
    <label for="producto">Producto:</label>
      <select name="producto" id="producto">
        <option value="Polo"> Polo </option>
        <option value="Quicksilver"> Quicksilver </option>
        <option value="Tommy"> Tommy </option>
      </select >

  <input type="submit" value="Enviar" />
  </form >
```

Método Post

Al contrario de GET, POST solo funciona enviando la información mediante formularios y no la muestra en el URL. La información que es enviada por POST va en el cuerpo de la solicitud HTTP y esta puede ir o no encriptada. Veamos un ejemplo de como manejar la información enviada por este método en PHP.

Primero creamos el formulario indicando que el método es POST.

```
<form id=”datos” name=”datos” action="pagina.php" method="POST" >
    <label for="usuario">Usuario: </label>
   <input type="text" name="usuario" id=”usuario”/>
   <label for="clave">Clave </label>
   <input type="password" name="clave" id=”clave”/>
   <label for="edad">Edad </label>
  <input type="number" name="edad" id=”edad”/>
  <input type="submit" value="Enviar" />
</form >
```

Nota: El campo edad es del tipo number para que el usuario no pueda incluir caracteres

Como podemos ver hay un campo contraseña, el cual sería muy inseguro pasar por GET y que alguien lo lograra ver en el URL.

Pagina.php

```
$usuario = $_POST['usuario'];
$clave = $_POST['clave'];
$edad = (int)$_POST['edad'];
echo "Usuario:  $usuario - clave: $clave - edad: $edad”;
var_dump ($edad);
```

Importante: Se ha transformado la información del campo edad en Entero (int) porque toda la información de un formulario son cadena de texto

Como podemos ver obtenemos los datos de la misma manera que con el método GET, solo que ahora la variable se llama $_POST.

Ejercicios

- ✓ **Ejercicio 1:** Realiza una copia de los cuatro ejercicios de variables e introduce los datos a través de formularios. Utiliza Fichero css externo para dar estilo a los formularios y utiliza tipos de campos correcto (text, number, etc)

4 Estructuras de control

Tabla de operadores

Los operadores de PHP son similares a los de cualquier otro lenguaje de programación. Estos son los operadores que se pueden aplicar tanto a las variables como a las constantes numéricas:

Operador	Ejemplo	Descripción
+	20 + $x	suma dos números
-	$a - $b	resta dos números
*	10 * 7	multiplica dos números
/	$altura / 2	divide dos números
%	5 % 2	devuelve el resto de la división entera
++	$a++	incrementa en 1 el valor de la variable
- -	$a- -	decrementa en 1 el valor de la variable
==	$a == $b	$a es igual $b (aunque sean de diferente tipo)
===	$a === $b	$a es igual $b (y además son del mismo tipo)
!=	$a != $b	$a es distinto $b
<	$a < $b	$a es menor que $b
>	$a > $b	$a es mayor que $b
<=	$a <= $b	$a es menor o igual que $b
>=	$a >= $b	$a es mayor o igual que $b
<=>	$a <=> $b	-1 si $a es menor, 0 si son iguales y 1 si $b es menor
&& o and	($a>$b) && ($a>$c)	Devuelve verdadero cuando ambas condiciones son ciertas
|| o or	($a>$b)|| ($a>$c)	Devuelve verdadero cuando alguna condición es cierta
!	! ($a>$b)	Niega el valor de la expresión

Sentencias Condicionales

Las sentencias condicionales son el núcleo para la toma de decisiones en los scripts de PHP. Estas sentencias básicamente controlan si parte de un código es ejecutado o no dependiendo del valor (True o False) que devuelve de una expresión que es evaluada. Visto de otra manera, estas sentencias dicen que camino se debe tomar a la hora de ejecutar el código.

Sentencia If

El bloque mas básico de un código condicional es el if. La primera línea de esta sentencia consiste en la palabra if seguida por la expresión que será evaluada entre paréntesis.

```
<?php
$variable = 1;
if ($variable < 2)
{
    // instrucciones que seran ejecutadas si la condición se cumple
```

```
    echo 'El valor de mi variable es menor que 2';
    $variable++;
}
?>
```

Sentencia If .. Else

La sentencia if anterior nos permite especificar que hacer si la expresión que se evalúa es verdadera. Sin embargo, no nos permite especificar que hacer cuando la expresión es evaluada como falsa. Aquí es donde aparece la sentencia if… else.

```
<?php
  $edad = 19;
  if( $edad >= 18){
    echo "Eres mayor de edad!";
  }else{
    echo "Eres menor de edad!";
  }
?>
```

Como se puede ver en el anterior ejemplo el código que sigue la sentencia if se ejecuta si la expresión que se evaluó es True, en cambio si la expresión fue False entonces se ejecuta el código que esta después de la sentencia else.

```
<?php
$usuario = "supervisor";
if ($usuario == "admin")
{
    echo 'Hola Admin, tiene todos los permisos';
}
else if ($usuario == "supervisor")
{
    echo 'Hola Supervisor';
}
else
{
    echo 'Hola ' . $usuario ;
}
?>
```

IF…. ELSE Abreviado

```
(condición) ? expresión1:expresión2
```

El funcionamiento es el siguiente: se evalúa la condición; si la condición es verdadera, el resultado que se devuelve es `expresión1` y si la condición es falsa, se devuelve `expresión2`.

```
<!DOCTYPE  html>
<html>
<head>
<meta  charset="UTF-8">
<title></title>
</head>
```

```
<body>
<?php
        $x = (20 > 6) ? 11 : 22;
        echo $x;
?>
</body>
</html>
```

Sentencia Switch Case

La sentencia if .. else funciona bien si queremos evaluar pocas opciones, pero cuando la cantidad de posibilidades se incrementa no es muy practico utilizar esta vía. En estos casos es cuando aparece la sentencia switch case, la cual se define de la siguiente manera.

```
switch ($variable)
{
    case "valor1" :
     Codigo PHP
    break;

    case "valor2" :
     Codigo PHP
    break;

    case "valor3" :
      Codigo PHP
    break;

    default :
     Codigo PHP
    break;
}
```

Pueden haber toda la cantidad de sentencias case que sean necesarias para comparar todas las opciones que se necesiten. Cuando una coincidencia es encontrada el código que encuentra justo después del case es ejecutado hasta donde se encuentre el break. La sentencia break; es muy importante ya que sin esta todos los cases siguientes se ejecutarían también. La sentencia default especifica la acción a ejecutar en caso que de que ninguno de los case se hayan ejecutado.

```
<?php
$carro = "Edge";
// La comparación en el switch verifica que la variable se igual a alguno de los datos de los case
switch ($carro)
{
    case "Caravan" :
    echo "Construido por Dodge";
    break;

    case "Supra" :
    echo "Construido por Toyota";
    break;

    case "Edge" :
```

```
    // En este caso se ejecutara este código
    echo "Construido por Ford";
    break;

    default  :
    echo "Modelo desconocido ";
    break;
}
?>
```

Sentencias de Iteración

Los bucles son la principal manera de indicarle al interprete de PHP que ejecute una tarea varias veces hasta que una condición se cumpla

Bucles For

```
for ( inicializador ; expresion condicional ; expresion bucle )
{
    // Código PHP que se va a repetir
}
```

El inicializador es una variable numérica que es puesta con el valor en donde se desea comenzar, normalmente se utiliza $i = 0. La expresión condicional especifica la condición que se debe aprobar para que el ciclo continúe, por ejemplo $i < 1000. Mientras $i sea menor que 1000 entonces el ciclo continuara repitiendose. Por último viene la expresión bucle, la cual especifica la acción a realizar con la variable $i. Por ejemplo, incrementar en 1 $i++.
Cuando juntamos todo esto podemos crear un bucle for

```
<?php
$j = 10;
for ($i=0; $i<10; $i++)
{
//sumar el valor de j consigo mismo 10 veces
 $j += $j;//Es lo mismo que $j=$j+$j;
}
Echo “$j”;
?>
```

Bucles While

El bucle for que vimos anteriormente funciona muy bien cuando sabemos la cantidad de veces que necesitamos repetir el código con anterioridad. En muchas ocaciones será necesario repetir un código pero sin saber cuando la condición de parada será cumplida, para estos casos esta el bucle while. Basicamente, el bucle while repite el código hasta que una cierta condición se cumpla. La sintaxis de este bucle es la siguiente

```
<?php
while (condicion)
{
```

```
    // PHP Código que se repite
}
?>
```

En el siguiente código la condición es una expresión que puede devolver True o False y podemos ver que el código que se repetirá será $variable = $variable + $j;.

```
<?php
$variable = 0;
while ($variable <= 100 )
{
//Muestra por pantala los números de 10 en 10 hasta que llegue al 100

    $variable = $variable + 10;
   Echo "$variable";
}
?>
```

Bucles Do .. While

Podemos pensar en este bucle como un while invertido. El bucle while primero evalúa la expresión y luego ejecuta el código dependiendo del resultado. Si la expresión que se evaluó retorna False la primera vez entonces el código nunca se ejecuta. En cambio el do .. while primero ejecuta el código y luego evalúa la expresión, en caso de que sea True entonces se vuelve a ejecutar y asi hasta que la expresión sea False.

```
<?php
do
{
   // Código PHP que se repite
} while (condicion)
?>
```

En el siguiente ejemplo el bucle va a continuar hasta que $i sea igual a 0, pero el do .. while asegura que mínimo se ejecute una vez.

```
<?php
$i = 10;
do
{
 $i--;
} while ($i > 0)
?>
```

Ejercicios

Ejercicio 1: Realiza un programa que pida una hora por teclado y que muestre luego buenos días, buenas tardes o buenas noches según la hora. Se utilizarán los tramos de 6 a 12, de 13 a 20

y de 21 a 5. respectivamente. Sólo se tienen en cuenta las horas, los minutos no se deben introducir por teclado.

Ejercicio 2: Vamos a ampliar uno de los ejercicios de la relación anterior para considerar las horas extras. Escribe un programa que calcule el salario semanal de un trabajador teniendo en cuenta que las horas ordinarias (40 primeras horas de trabajo) se pagan a 12 euros la hora. A partir de la hora 41, se pagan a 16 euros la hora

Ejercicio 3: Pide por teclado 3 notas y calcula la nota media y diga si tiene insuficiente , suficiente, notable o sobresaliente.

Ejercicio 4 Escribe un programa que ordene 3 número introducido por teclado.

Ejercicio 5: Introduce un número (como máximo 4 cifras). Muestra por pantalla la primera cifra, la última y di si es par o impar. Nota: Formulario utiliza min y max.

Ejercicio 6 : Muestra los múltiplos de 5 entre 0 y 100

Ejercicio 7: Haz una tabla HTML donde muestre la tabla de multiplicar del 1 al 10. La tabla tendrá un encabezado (<th></th>)en la primera fila (Tabla 1, Tabla 2,….).

Ejercicio 8: Escribe un programa que muestre los n primeros términos de la serie de Fibonacci. El primer término de la serie de Fibonacci es 0, el segundo es 1 y el resto se calcula sumando los dos anteriores, por lo que tendríamos que los términos son 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144… El número n se debe introducir por teclado.

Ejercicio 9: Escribe un programa que diga si un número introducido por teclado es o no primo. Un número primo es aquel que sólo es divisible entre él mismo y la unidad.

Ejercicio 10: Introduce dos número por teclado e imprime los número impares que hay entre ellos. Controla que el primer número sea menor que el segundo

5 Includes y páginas reiteradas

En PHP es muy común separar el código en diferentes archivos y luego ir llamándolos a unos u a oros según sea necesario: Para ellos se pueden utilizar las siguientes funciones

Función	Definición
Require(ruta/archivo.php);	Establece que el código del archivo invocado es obligatorio para el funcionamiento del programa. Se dentendrá el programa PHP si el código no se encuentra
Require_once(ruta/archivo.php);	Igual que la función require impidiendo cargar el fichero más de una vez.
Include(ruta/archivo.php);	Si el código que invocamos no se encuentra, el programa seguirá ejecutándose
Include_once(ruta/archivo.php);	Igual que la función include impidiendo cargar el fichero más de una vez.

Nota: La función **require_once** es la más utilizada

Carga reiterada de una página.

En el próximo ejemplo adivina un número entre 1 y 50 la página del formulario HTML y la página PHP son la misma que se irá repitiendo hasta que se acaben las oportunidades o se acierte el número.

index.php

```
<!DOCTYPE  html>
<html>
<head>
<meta  charset="UTF-8">
</head>
<body>
<?php
if (!isset ($_POST['numeroIntroducido'])){
       $oportunidades =5;
      $numerointroducido=0;
      $numeroSecreto =  24;
   echo"<h1> Bienvenido al juego adivina un número. Tienes $oportunidades oportunidades para acertar un número entre 1 y 50</h1>";
   require_once 'formadivina.php';


}else {

          $oportunidades =  (int)$_POST['oportunidades'];
   $numeroIntroducido =  (int)$_POST['numeroIntroducido'];

if ($numeroIntroducido  ==  $numeroSecreto)  {
   echo   "<h1>Enhorabuena, has acertado  el  número.</h1>";
}  elseif ($oportunidades  ==  0)  {
   echo   "<h1>Lo  siento, has agotado todas tus  oportunidades. Has  perdido</h1>";

}else {
```

```
    $oportunidades--;
    if ($numeroSecreto > $numeroIntroducido){
      echo "<h1>El número que estoy pensando es mayor que el número que has introducido.</h1>";
    }else {
      echo "<h1>El número que estoy pensando es menor que el número que has introducido</h1>";
   }

   require_once 'formadivina.php';
  }}
?>

</body>
</html>
```

Formadivina.php

```
<?php
echo "<form action='index.php' method='post'>
   <input type='number' name='numeroIntroducido'>
   <input type='hidden' name='oportunidades' value=' $oportunidades '>
   <input type='submit' value='Continuar'>
</form>
<p>Te quedan $oportunidades oportunidades para adivinar el número.<br/> Introduce un número del 0 al 100</p>"
?>
```

Ejercicios

- ✓ **Ejercicio 1:** Utilizando el ejercicio Web del tema 1, separa la cabera y el menú en un fichero diferente y utiliza Require_once para añadir el código en la página principal y el formulario. Los ficheros html tendrán extensión php
- ✓ **Ejercicio 2:** Realiza el control de acceso a una caja fuerte. La combinación será un número de 4 cifras. El programa nos pedirá la combinación para abrirla. Si no acertamos, se nos mostrará el mensaje "Lo siento, esa no es la combinación" y si acertamos se nos dirá "La caja fuerte se ha abierto satisfactoriamente". Tendremos cuatro oportunidades para abrir la caja fuerte.
- ✓ **Ejercicio 3:** Escribe un programa que calcule la media de un conjunto de números positivos introducidos por teclado. A priori, el programa no sabe cuántos números se introducirán. El usuario indicará que ha terminado de introducir los datos cuando meta un número negativo.

6 Funciones

Introducción a la funciones (Bibliotecas)

Las funciones son un trozo de código que realiza una tarea muy concreta y que se puede incluir en cualquier programa cuando hace falta resolver esa tarea. A las funciones se les pueden pasar argumentos o parámetros de ser necesario para que utilicen sus valores para realizar alguna operación y retorna algún valor al final de la ejecución.
En PHP existen **dos tipos de funciones**, las que PHP trae por defecto para que el programador las utilice y las que el programador crea desde cero dependiendo de sus necesidades.

Por lo general y salvo casos puntuales que van incluidas dentro de fichero principal, las funciones se suelen agrupar en ficheros independiente (**BIBLIOTECA DE FUNCIONES**). Estos ficheros de funciones se incluyen posteriormente en el programa principal mediante `include` o `require` seguido del nombre completo del fichero.

Veamos cómo utilizar la función **esPrimo()** desde un fichero independiente llamado **matemáticas.php**

```
<?php


function esPrimo($n)  {
$esPrimo =  true;


for ($i =  2;  $i <  $n;  $i++)  {
if ($n  %   $i ==  0)  {
$esPrimo =  false;
}
}

// El 0  y el 1  no  se consideran primos
if (($n  ==  0)  || ($n  ==  1))  {
$esPrimo =  false;
}

return $esPrimo;
}
?>
```

Programa principal

Index.php

```
<!DOCTYPE  html>
<html>
<head>
<meta  charset="UTF-8">
<title></title>
</head>
<body>
<?php
```

```
// Carga  las  funciones matemáticas
Require_once 'matematicas.php';
if (!isset($_POST['numero'])) {
?>
<p>Introduzca un  número para  saber  si es  primo o  no.</p>
<form  action=index.php  method="post">
      <input  type="number"  name="numero" min="0" autofocus><br/>
      <input type="submit"  value="Aceptar">
</form>
<?php
} else {
// A pesar del tipo number realizamos tambien un comprobación por PHP para ver si es número
If (is_numeric(($_POST['numero'])){
  $numero=(int)$_POST['numero'];
   if (esPrimo($numero)) {
                echo  "El $numero  es primo.";
     } else {
          echo  "El $numero  no  es primo.";}
}else{
  Echo “el valor no es número”;
}
?>
</body>
</html>
```

Funciones predefinidas

Vamos a ver alguna de las funciones predefinidas más importantes

Nombre Función	Explicación
Var_dump($variable);	Debugger y ver información variable
Time();	Fecha
Sqrt (int);	Raiz cuadrada
Rand(int1,int2)	Función aleatorio
Pi();	Valor pi
Round(Float,decimales);	Redondear
Gettype($variable);	Obtener el tipo de una variable
Is_string($variable)/Is_float($variable);	Comprobar el tipo
Is_numeric($variable)/Is_int($variable)	Comprueba si es un número entero, utilizamos is_numeric cuando el valor es introducido por un formulario.
Isset($variable);	Comprueba si existe la variable
Empty($variable)	Comprueba si esta vacio
Trim($variable);	Limpiar espacios
Unset($variable);	Eliminar variable
Strlen($cadena);	Contar caracteres
Strpos($cadena,letra)	Buscar dentro cadena
Str_replace (“palabra1”,”palabra2”,$cadena)	Reemplazar dentro cadena
Strtolower($cadena)/ strtoupper($cadena)	Convertir Minúsculas y mayúsculas

Ejercicios

- ✓ **Biblioteca : Crea** una biblioteca de funciones matemáticas que contenga las siguientes funciones. Recuerda que puedes usar unas dentro de otras si es necesario.

    1. esCapicua: Devuelve verdadero si el número que se pasa como parámetro es capicúa y falso en caso contrario.
    2. esPrimo: Devuelve verdadero si el número que se pasa como parámetro es primo y falso en caso contrario.
    3. siguientePrimo: Devuelve el menor primo que es mayor al número que se pasa como parámetro.
    4. potencia: Dada una base y un exponente devuelve la potencia.
    5. digitos: Cuenta el número de dígitos de un número entero.
    6. voltea: Le da la vuelta a un número.
    7. digitoN: Devuelve el dígito que está en la posición n de un número entero. Se empieza contando por el 0 y de izquierda a derecha.
    8. posicionDeDigito: Da la posición de la primera ocurrencia de un dígito dentro de un número entero. Si no se encuentra, devuelve -1.
    9. quitaPorDetras: Le quita a un número n dígitos por detrás (por la derecha).
    10. quitaPorDelante: Le quita a un número n dígitos por delante (por la izquierda).
    11. pegaPorDetras: Añade un dígito a un número por detrás.
    12. pegaPorDelante: Añade un dígito a un número por delante.
    13. trozoDeNumero: Toma como parámetros las posiciones inicial y final dentro de un número y devuelve el trozo correspondiente.
    14. juntaNumeros: Pega dos números para formar uno.

Utilizando la biblioteca de funciones resuelve

- ✓ **Ejercicio 1:** Muestra los números primos que hay entre 1 y 1000.
- ✓ **Ejercicio 2:** Muestra los números capicúa que hay entre 1 y 99999.
- ✓ **Ejercicio 3:** Escribe un programa que pase de binario a decimal.
- ✓ **Ejercicio 4:** Escribe un programa que pase de decimal a binario.

7 Arrays

Crear y trabajar con Arrays

Un array es un tipo de dato capaz de almacenar múltiples valores. Se utiliza cuando tenemos muchos datos parecidos, por ejemplo, para almacenar la temperatura media diaria en Cádiz (guardados de uno en uno), películas o los colores (en una linea). La posición inicial de los array es 0.

```
<!DOCTYPE html>
<html lang=”es”>
<head>
        <meta charset="UTF-8">
</head>
<body>
<?php
        //declaramos 3 arrays de manera diferente
        $temp[0] = 16;
        $temp[1] = 15;
        $temp[2] = 17;
        $temp[3] = 15;
        $temp[4] = 16;
        $color = ["verde", "amarillo", "rojo", "azul", "blanco", "gris"];
        $peliculas= array (“La llegada”, “Star Wars”, “Batman el caballero oscuro”);
        // Imprimimos por pantalla algún dato de los array
        echo "<p>La temperatura en Málaga el cuarto día fue de $temp[3] ºC</p>";
        echo ‘<p>Mañana me pongo una camiseta de color ’. $color[0].’</p>’;
        echo ‘ <p> Mi película preferida es ‘. $peliculas[0].’</p>’;
        // Otra manera de insertar un valor en la última posición del array $color
         $color[]="negro";
          echo "Mañana me pongo una camiseta de color ", $color[6], ".";

?>

</body>
</html>
```

Recorrer un Arrays (foreach o for)

Foreach es el iterador más utilizado se para recorrer todos los elementos de un array sin que tengamos que preocuparnos por los índices ni por el tamaño del array.
Como puedes ver en los ejemplos anteriores, no es necesario definir previamente un array antes de utilizarlo. Tampoco hay límite en cuanto al número de elementos que se pueden añadir al mismo. No obstante, se pueden **crear arrays de tamaño fijo con SplFixedArray** que son más eficientes en cuanto a uso de la memoria y más rápidas en las operaciones de lectura y escritura.

La función **var_dump()** se utiliza para mostrar el tipo y el valor de un dato, en este caso muestra los tipos y valores de cada uno de los elementos del array.

```
<!DOCTYPE  html>
<html>
<head>
<meta  charset="UTF-8">
</head>
<body>
<?php
// Declaramos un array con 10 posiciones
$a  =  new  SplFixedArray(10);
$a[0]  =  843;
$a[2]  =  11;
$a[6]  =  1372;
// Los   valores del array que   no  se han  inicializado son   NULL
foreach ($a  as $elemento)  {
     var_dump($elemento);
echo   "<br/>";
// Crear un array n con números aleatorios entre 0  y 10  (ambos incluidos)
for ($i =  0;  $i <  10;  $i++)  {
     $n[$i]  =  rand(0,  10);
}
// Mostramos los elementos con foreach
foreach ($n  as $elemento)  {
    echo   $elemento. "<br/>";}
?>
</body>
</html>
```

```
<!DOCTYPE  html>
<html lang=”es”>
<head>
<meta  charset="UTF-8">
</head>
<body>
<?php
$cajonDeSastre =  [30,  -7,  "Me  gusta el queso",  56,  "¡eh!",  237];
// Mostrar el array utilizando foreach
foreach ($cajonDeSastre  as $cosa)  {
echo   "$cosa<br/>";}
// Mostrar el mismo array utilizando for
echo "<hr/>";
for ($i=0;$i< count($cajonDeSastre);$i++){
      echo   "$cajonDeSastre[$i]<br/>";}
?>
</body>
</html>
```

Arrays asociativos

En un array asociativo se pueden utilizar índices que no son numéricos, a modo de claves. Veamos un ejemplo de un array asociativo que almacena alturas (en centímetros) y que como índice o clave utiliza nombres. En este ejemplo vamos a ver tres formas de asignar los valores

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="UTF-8">
 </head>
 <body>
  <?php
  // Tres maneras de asignar valores a un Array asociativo
  //Asignando los valores con array(valores);
  $estatura = array("Rosa" => 168, "Ignacio" => 175, "Daniel" => 172, "Rubén" =>
182);
    echo "La estatura de Daniel es ". $estatura['Daniel'] . " cm<br/>";
    //Asignando los valores individualmente
    $estatura2['Rosa'] = 168;
    $estatura2['Ignacio'] = 175;
    $estatura2['Daniel'] = 172;
    $estatura2['Rubén'] = 182;
    //Mostramos por pantalla una lista de las alturas (Sin nombre)
    foreach ($estatura2 as $altura){
      echo "$altura cm<br/>";
    }
    // Asignando con corchetes
    $estatura3 =  [ "Rosa"  =>  168, "Ignacio" =>  175, "Daniel" =>  172, "Rubén" =>
182];
    //Mostramos por pantalla todos los valores con sus claves
    foreach ($estatura3 as $persona => $altura2) {
       echo "La estatura de $persona es $altura2 cm<br/>";}
    ?>
 </body>
</html>
```

Arrays bidimensionales

Un array bidimensional utiliza dos índices para localizar cada elemento. Podemos ver este tipo de datos como un array que, a su vez, contiene otros arrays. En el siguiente ejemplo se define un array con 4 elementos que, a su vez, son un array asociativo cada uno de ellos.

```
<!DOCTYPE html>
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
 </head>
 <body>
```

```
<?php
  $persona = array (
    array( "nombre" => "Rosa", "estatura" => 168, "sexo" => "F"),
    array( "nombre" => "Ignacio", "estatura" => 175, "sexo" => "M"),
    array( "nombre" => "Daniel", "estatura" => 172, "sexo" => "M"),
    array( "nombre" => "Rubén", "estatura" => 182, "sexo" => "M")
  );

  echo "<b>DATOS SOBRE EL PERSONAL<b><br><hr>";

  $numPersonas = count($persona);

  for ($i = 0; $i < $numPersonas; $i++) {
    echo "Nombre: <b/>". $persona[$i]['nombre']. "</b><br>";
    echo "Estatura: <b/>". $persona[$i]['estatura']. " cm</b><br>";
    echo "Sexo: <b/>". $persona[$i]['sexo']. "</b><br/><hr>";
  }
  // Utilizando un doble foreach mostramos los datos del array bidemencional
  foreach ($persona as $array){
     foreach ($array as $dato => $valor) {
       echo "$dato : <b/>". $valor. "</b><br/>";  }
     echo '<hr/>';
     }
  ?>
 </body>
</html>
```

Observa que la función **count()** permite saber el número de elementos de un array.

Cómo recoger datos para un array mediante un formulario

Imagina que quieres pedir diez números por teclado y quieres guardar esos números en un array. Se puede pedir un número mediante un formulario, a continuación, el siguiente, luego el siguiente, etc. pero ¿cómo enviarlos? Hay un truco muy sencillo. Se pueden ir concatenando valores en una cadena de caracteres y el resultado de esa concatenación se puede reenviar una y otra vez en un formulario como campo oculto. Por último, se puede utilizar la función explode() para convertir la cadena de caracteres en un array.

Es importante señalar que los valores que se van concatenando deben tener algún tipo de separación dentro de la cadena, por ejemplo un espacio en blanco.

A continuación se muestra un ejemplo completo:

```
<!DOCTYPE html>
<html>
 <head>
   <meta charset="UTF-8">
 </head>
 <body>
   <?php
    if (!isset($_GET['n'])) {
      $contadorNumeros = 0;
```

```
      $numeroTexto = "";
      $n="";
    }else{
       $contadorNumeros = $_GET['contadorNumeros'];
      $numeroTexto = $_GET['numeroTexto'];
      $n=$_GET['n'];
    }

    // Muestra los números introducidos
    if ($contadorNumeros == 6) {
      $numeroTexto = $numeroTexto . " " . $n; // añade el último número leído
      $numeroTexto = substr($numeroTexto, 2); // quita los dos primeros
                                              // espacios de la cadena
      $numero = explode(" ", $numeroTexto);
      echo "Los números introducidos son: ";
      foreach ($numero as $num) {
        echo $num, "  ";
      }
    }

    // Pide número y añade el actual a la cadena
    if (($contadorNumeros < 6) || (!isset($_GET['n']))) {
      ?>
    <form action="#" method="GET">
        Introduzca un número:
        <input type="number" name ="n" autofocus>
        <input type="hidden" name="contadorNumeros" value="<?=
++$contadorNumeros ?>">
        <input type="hidden" name="numeroTexto" value="<?= $numeroTexto . " " . $n
?>">
        <input type="submit" value="OK">
      </form>
      <?php
    }
   ?>
  </body>
</html>
```

Funciones predefinidas Array

Vamos a ver alguna de las funciones predefinidas más importantes

Nombre Función	Explicación
Sort($array);	Ordenar Array
Array_push($array,elemento)	Añadir uno o más elementos al final del array
Array_unshift()	Añadir uno o más elementos al principio del array
Array_pop(int); **Unset($Array[indice]))**	Elimina un elemento del final del array
Array_Shift($array	Elimina un elemento del principio del array
$indice=array_rand($array);	Elige un índice al azar
Array_reverse($array)	Ordenar de manera inversa los elementos
Array_search(num,$array)	Buscar un elemento
Count($array);	Longitud array
in_array($palabra,$arraypalabras)	Comprueba si existe la palabra dentro del array

Ejercicios

- ✓ **Ejercicio 1:** Haz un programa con un array de 8 números enteros aleatorios (0,10) y haz lo siguientes
  - Haz una función mostrarArray($Array).
  - Haz una función que devuelva un array sin el último elemento del array (No utilices funciones predefinidas). Compara el resultado de tu función y la función predefinida
  - Ordenarlo y mostrarlo
  - Mostrar su longitud
  - Buscar algún elemento y mostrar el índice
  - Voltea el Array
  - Busca algún elemento que nos introduzcan por la url
- ✓ **Ejercicio 2:** Añade valores (Elemento-1,Elemento-2,etc) a un array mientras su longitud sea menor de 120 y mostrarlo por pantalla
- ✓ **Ejercicio 3:** Crea variables tipo array, string, integer, booleano, imprime un mensaje por pantalla dependiendo el tipo de variable.
- ✓ **Ejercicio 4:** Crea un array asociativo donde cada elemento es un array con el contenido de la tabla**.**

Accion	Aventura	Deporte
GTA 5	Assasin Creed	FIFA
Call Of duty	Tomb Raider	PES
PUGB	Last of us	Moto G

Haz una tabla HTML con los datos del array

- ✓ **Ejercicio 5:** Escribe un programa que pida 10 números por teclado y los almacene en un array. Muestra los valores máximo y mínimo

- ✓ **Ejercicio 6:** Crea un mini-diccionario español-inglés que contenga, al menos, 10 palabras (con su traducción). Utiliza un array asociativo para almacenar las parejas de palabras. El programa pedirá una palabra en español y dará la correspondiente traducción en inglés.

Pista: Utiliza la función in_array($palabra,$arraypalabras) para comprobar si la palabra está dentro. Anteriormente tendrás que crear un array simple con las claves (palabras).

- ✓ **Ejercicio 7:** Realiza un programa que escoja al azar 5 palabras en español del mini-diccionario del ejercicio anterior. El programa pedirá que el usuario teclee la traducción al inglés de cada una de las palabras y comprobará si son correctas. Al final, el programa deberá mostrar cuántas respuestas son válidas y cuántas erróneas.