

Programación de aplicaciones web

DESARROLLO WEB EN ENTORNO SERVIDOR. UNIDAD 3

2º DESARROLLO DE APLICACIONES WEB. ARCIPRESTE DE HITA. 2025/2026

AARÓN MONTALVO

3. Programación de aplicaciones web.

Criterios de evaluación

- a) Se han utilizado mecanismos de decisión en la creación de bloques de sentencias.
- b) Se han utilizado bucles y se ha verificado su funcionamiento.
- c) Se han utilizado matrices (*arrays*) para almacenar y recuperar conjuntos de datos.
- d) Se han creado y utilizado funciones.
- e) Se han utilizado formularios web para interactuar con el usuario del navegador web.
- f) Se han empleado métodos para recuperar la información introducida en el formulario.
- g) Se han añadido comentarios al código.

3. Programación de aplicaciones web.

Contenidos

3.1. Formularios

3.2. Sentencias de control

3.3. Matrices

3.4. Cadenas de caracteres

3.5. Funciones definidas por el usuario

3.1. Formularios

Las variables superglobales están presentes en todos los ámbitos de un programa PHP y recogen información valiosa para el programa. Tres de ellas recogen información de formularios HTML.

- `$_GET['nombre_etiqueta']`: Accede a los datos mandados desde los formularios mediante el método GET tras un submit.
- `$_POST['nombre_etiqueta']`: Accede a los datos mandados desde los formularios mediante el método POST tras un submit.
- `$_REQUEST['nombre_etiqueta']`: Accede a los datos mandados desde los formularios mediante cualquiera de los métodos GET o POST tras de un submit.

3.1. Formularios

Estos métodos funcionan correctamente para controles de tipo text, number, select y radio.

```
$nombre = $_REQUEST['nombre'];  
$apellidos = $_REQUEST['apellidos'];  
echo "Bienvenido, $nombre $apellidos";
```

3.1. Formularios

Para checkbox, los valores sólo se envían en caso de ser marcados

- Es necesario hacer una comprobación antes de recoger el valor mediante `isset()` o `filter_has_var()`.

```
$check = isset($_REQUEST['cb']) ? $_REQUEST['cb'] : "No pulsado";
```

```
$check = $_REQUEST['cb'] ?? "No pulsado"; //Fusión de nulo
```

```
$check = "No pulsado";
```

```
if (filter_has_var(INPUT_GET, 'cb')) $check = $_GET['cb'];
```

3.1. Formularios.

Subida de archivos

PHP permite subir archivos utilizando el método POST.

Para ello, debe crearse un formulario que use el método POST y que contenga un control de tipo archivo "file".

- El formulario debe estar codificado con "multipart/form-data".

La información de todos los archivos subidos se guardará en la variable superglobal `$_FILES`.

- El índice de cada archivo será el atributo de nombre del control de tipo archivo.
- <https://www.php.net/manual/es/features.file-upload.post-method.php>

3.1. Formularios.

Subida de archivos

Ejemplo (1/2): Formulario

```
<form enctype="multipart/form-data" action="ut02_files.php"
    method="POST">
    <input type="hidden" name="MAX_FILE_SIZE" value="30000">
    <input name="sube" type="file">
    <button type="submit">Importar archivo</button>
</form>
```


3.1. Formularios.

Subida de archivos

Ejemplo (2/2): Procesamiento en servidor

```
$dir_subidas = $_SERVER["DOCUMENT_ROOT"].'/dwes/uploads/';
$fn = $dir_subidas . basename($_FILES['sube']['name']);
echo '<pre>';
if (move_uploaded_file($_FILES['sube']['tmp_name'], $fn)) {
    echo "El archivo es válido y se subió con éxito.\n";
} else {
    echo "Error al subir el archivo es válido.\n";
}
echo 'Más información:'.print_r($_FILES, true).'
```

Actividad 3.1

- a. Crea un formulario HTML que solicite al usuario que introduzca una serie de valores: nombre, apellidos, dirección y teléfono, y envía dichos datos a un archivo PHP que recogerá los valores y los mostrará en pantalla en una tabla.
- b. Crea un formulario HTML que recoja dos valores numéricos a través de cuadros de texto y permita seleccionar una operación matemática entre: suma, resta, multiplicación, división y módulo. Envía los valores a un archivo PHP que realizará la operación indicada entre los dos números y mostrará en pantalla el resultado.

Actividad 3.2

- a. Crea un formulario HTML que recoja los datos de un pedido para una tienda online que vende 3 tipos de artículos. El formulario deberá recoger el número de unidades solicitadas de cada tipo. Recoge los valores y genera una factura en la que se indiquen las unidades vendidas de cada uno de los artículos, su precio unitario (5.99€, 12.49€ y 19.99€ respectivamente), subtotal por artículo e importe total aplicando un 20% de IVA. Usa alguna función que te permita incluir la fecha de emisión de la factura.

Artículo	Precio	Unidades	Subtotal
Artículo A	5.99€	X	
Artículo B	12.49€	Y	
Artículo C	19.99€	Z	
			IVA (20%)
			TOTAL

- b. Modifica la tabla anterior de manera que se haga un descuento en función del número total de unidades vendidas. En la factura se deberá mostrar tanto el porcentaje de descuento como su valor en euros

Nº Unidades	Descuento
Inferior a 5	Sin descuento
Entre 5 y 10	5%
Entre 11 y 20	10%
Superior a 20	25%

Artículo	Precio	Unidades	Subtotal
Artículo A	5.99€	X	
Artículo B	12.49€	Y	
Artículo C	19.99€	Z	
			Dto. (xx%)
			IVA (20%)
			TOTAL

Actividad 3.3

Crea un formulario HTML que recoja un texto a través de un cuadro de texto y las características del borde: estilo (liso, doble, punteado u oculto), tamaño en pixeles (entre 0 y 10) y color (negro, rojo, azul o amarillo).

Envía los valores a un archivo PHP y muestra el texto en un cuadro con las características establecidas en el formulario.

Formatos en Texto

Texto:

Características del borde

Tamaño:

Estilo:

Oculto ▼

Color:

Negro ▼

Enviar

Actividad 3.4

- a. Crea una carpeta "inc" en tu servidor web y guarda en ella los archivos "header.php" y "footer.php":
 - "header.php" establecerá los estilos CSS a utilizar en las páginas del sitio, y la cabecera que tendrán todas esas páginas que deberá estar formada por un banner con el nombre del sitio y su logo, y un menú horizontal con varias opciones: inicio, productos, contacto, etc.
 - "footer.php" contendrá un pie de página con el nombre de la empresa y su contacto.
- b. Genera un archivo PHP que utilice estos dos archivos, y muestre algo en el cuerpo de la página.



Actividad 3.5

- a. Modifica el archivo "php.ini" para que se carguen automáticamente los archivos "header.php" y "footer.php" del ejercicio anterior.
- b. **Antes de modificar el archivo "php.ini" haz una copia de seguridad, y ten en cuenta que una vez modificado deberás reiniciar Apache para que los cambios surtan efecto.**
- c. Crea un archivo PHP para comprobar que la configuración de "php.ini" funciona correctamente. Éste archivo sólo debe contener las etiquetas PHP y un echo en su interior para mostrar un mensaje en pantalla, pero ninguna etiqueta de estructura html: HTML, HEAD, BODY, etc.
- d. Comprueba qué pasa si intentas cargar otro archivo PHP, y comprueba el código HTML que se ha generado.
- e. Devuelve "php.ini" a su estado original. Utiliza esa configuración sólo para los archivos de una carpeta. Comprueba que no es necesario reiniciar Apache para que los cambios

3.2. Sentencias de control

Todo *script* PHP está construido con una serie de sentencias.

Una sentencia puede ser una asignación, una llamada a función, un bucle, una sentencia condicional o incluso una sentencia vacía.

Las sentencias pueden agruparse en conjuntos, encapsulándolas entre corchetes.

- Un grupo de sentencias es una sentencia por sí misma también.

3.2. Sentencias de control. Condicionales. if/else

El constructo `if` permite la ejecución condicional de fragmentos de código.

`if (expr) statement`

La expresión se evalúa a su valor booleano. Si es cierta, PHP ejecutará la sentencia y si es falsa PHP la ignorará.

```
if ($nota >= 5) {  
    echo "<p style='color:blue'>Aprobado</p>";  
}
```


3.2. Sentencias de control. Condicionales. if/else

Se puede ejecutar una sentencia en caso que la expresión en la sentencia `if` se evalúe como falsa usando `else`.

```
if ($nota >= 5) {  
    echo "<p style='color:blue'>Aprobado</p>";  
} else {  
    echo "<p style='color:red'>Suspenso</p>";  
}
```

3.2. Sentencias de control. Condicionales. if/else

Se puede combinar `else` con `if` para que, en caso de que una expresión sea falsa, se evalúe una nueva expresión.

- Puede haber varios `elseif`, pero un único `else`.

```
if ($nota >= 9) {  
    echo "<p>Sobresaliente</p>";  
} elseif ($nota >= 5) {  
    echo "<p>Aprobado</p>";  
} else {  
    echo "<p>Suspenso</p>";  
}
```

3.2. Sentencias de control. Condicionales. switch

Es similar a una serie de sentencias `if` sobre la misma expresión.

```
switch ($nota) {  
    case 10: case 9:  
        echo "<p>Sobresaliente</p>"; break;  
    case 8: case 7: case 6: case 5:  
        echo "<p>Aprobado</p>"; break;  
    default:  
        echo "<p>Suspenso</p>";  
}
```

3.2. Sentencias de control.

Bucles. for

Evalúa una expresión al inicio del bucle, otra al inicio cada iteración y otra al final de cada iteración.

```
for (expr1; expr2; expr3) statement
```

Si la segunda expresión es falsa, se finaliza la ejecución del bucle.

```
for ($i=1; $i<=10; $i++) {  
    echo "<p>i = $i</p>";  
}
```

3.2. Sentencias de control.

Bucles. for con variables variables

Se puede utilizar un bucle for junto con las variables variables para procesar varios variables de nombre similar:

```
for ($i=1; $i<=$numnombres; $i++) {  
    $temp = "nombre$i";  
    echo "$temp<br/>";  
}
```

En el ejemplo cada variable se debería llamar: nombre1, nombre2, nombre3, etc. hasta el número indicado por numnombres.

3.2. Sentencias de control.

Bucles. foreach

Constructo que se usa para iterar sobre matrices y objetos.

`foreach (iterable_expression as $value) statement`

En cada iteración, el valor del elemento actual se asigna a `value` y el puntero interno avanza una posición.

```
$array = array(1, 2, 3, 4);  
foreach ($array as &$amp;valor) {  
    $valor = $valor * 2;  
    echo "<p>$valor</p>";  
}
```

3.2. Sentencias de control.

Bucles. foreach

También se pueden recoger las claves además de los valores.

```
foreach (iterable_expression as $key => $value) statement
```

Cada clave se recogerá en `key` y cada valor en `value`.

```
$array = array("A" => 1, "B" => 2, "C" => 3);  
foreach ($array as $clave => $valor) {  
    echo "<p>{$clave} => {$valor}</p>";  
}
```

3.2. Sentencias de control.

Bucles. while

Es el tipo más sencillo de bucle en PHP.

`while (expr) statement`

El valor de la expresión es verificado al inicio de cada bucle

- Si este valor cambia durante la ejecución, esta no se detendrá hasta el final de la iteración. Si la expresión es falsa al inicio, no se hará ninguna iteración.

```
$num = 1;
while($num<=10) {
    echo '<p>'.$num++.'</p>';
}
```


3.2. Sentencias de control.

Bucles. do-while

Similar al bucle `while`, pero la expresión se verifica al final de cada iteración en vez de al inicio.

Al menos se ejecutará una iteración.

```
$num = 100;  
do {  
    echo '<p>'.$num++.'</p>';  
    $num = $num - 10;  
} while($num < 10);
```

3.2. Sentencias de control.

Bucles. Sentencias de ruptura

`break`

Finaliza la ejecución de la estructura `switch`, `for`, `foreach`, `while`, o `do-while` en curso.

- Acepta un argumento numérico opcional para indicar de cuántas estructuras salir.

`continue`

Se salta el resto de la iteración actual del bucle y vuelve a evaluar la condición.

- También admite un argumento numérico opcional.

3.2. Sentencias de control.

Sintaxis alternativa

Existe una sintaxis alternativa para estas estructuras de control

Consiste en sustituir la llave de apertura ({) por dos puntos (:) y la llave de cierre (}) por la sentencia end correspondiente

- endif, endswitch, endwhile, endfor o endforeach.

```
if ( $nota == 0 ):
    echo "No ha introducido la nota";
else:
    echo "La nota es $nota";
endif;
```

Actividad 3.6

- a. Recoge dos valores numéricos (filas y columnas) a través de un formulario y crea una tabla con el número de filas y columnas indicadas, introduciendo en las celdas los "n" primeros números naturales.
 - $n = \text{filas} * \text{columnas}$.
- b. Crea un formulario para recoger la información personal de los alumnos de un instituto de secundaria: nombre, apellidos, teléfono, dirección, población, provincia, fecha de nacimiento y estudios (ESO, Bachillerato, Ciclo Formativo). Recoge toda la información y muéstrala dentro de una tabla utilizando sendos bucles.

Actividad 3.7

Realiza una aplicación que permita recoger un número entero y una función (opuesto, inverso, cuadrado, raíz cuadrada, sumatorio (n primeros números) o factorial) a través de un formulario y a continuación muestre en pantalla la función realizada y el resultado obtenido.

- Si la función no está definida para el valor entero suministrado, se deberá indicar con el texto: "No existe".
- El título del documento de salida deberá coincidir con la función realizada y no se podrán utilizar funciones matemáticas excepto para la raíz cuadrada.



Actividad 3.8.

Juego de dados

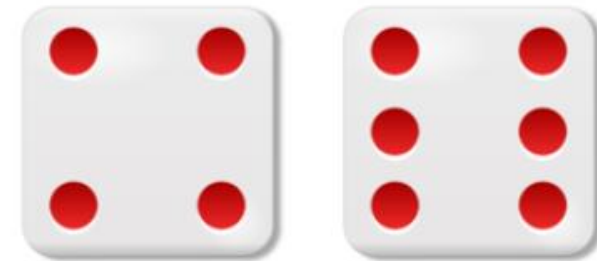
El jugador se presenta con su nombre y trata de adivinar lo que va a obtener al lanzar dos dados.

A continuación, lanzará los dados y si la suma de ambos coincide con el número introducido por el jugador, ganará la partida, en caso contrario gana la banca.

Para hacerlo un poco más atractivo puedes buscar imágenes de cada una de las caras de un dado para mostrar la tirada del jugador en pantalla.

Suerte, Oscar

Jugada: 10



!!! Enhorabuena Oscar, ha ganado !!!

Actividad 3.9

- a. Crea un archivo PHP que realice dos acciones:
 - si no se han capturado datos muestra un formulario solicitando nombre y apellido
 - si ya se han capturado esos datos muestra un mensaje de bienvenida.
- b. Crea un formulario en PHP que permita recoger los nombres de los alumnos de una clase (máximo 10) y a continuación recoge los valores y muéstralos en una tabla.
- c. Crea un formulario en PHP con varios controles y a continuación captura toda la información del formulario: nombres de los campos y valores y muéstralos en una tabla.

3.3. Matrices

Una matriz o array es una estructura que permite almacenar una colección de datos (elementos) de cualquiera de los tipos básicos (`int`, `double`, `string`, etc.), o incluso otra matriz (matriz multidimensional).

Para acceder a un elemento de la matriz se usa la notación `[]` o `{}` indistintamente junto a un índice, que puede ser

- un número: matriz indexada numéricamente
- una cadena de texto: matriz asociativa.

PHP permite que elementos de una misma matriz sean de tipos diferentes.

3.3. Matrices

Para poder imprimir información sobre una matriz de manera legible para humanos hay que usar la función `print_r()`.

```
print_r(mixed $expression, bool $return = false): mixed
```

Esta función también se puede usar con otros tipos como string, integer o float, pero al usarlo con array se mostrarán los datos en formato clave => valor.

- El segundo parámetro sirve para, en lugar de imprimir la información, devolverla para poder asignarla a una variable.

3.3. Matrices.

Indexada numéricamente

Cada elemento de la matriz es accedido a través de un **índice numérico que comienza en 0.**

```
$alumnos[0] = "Ana";  
$alumnos[1] = "Juan";  
$alumnos[2] = "Pedro";  
for ($i=0; $i<3; $i++) {  
    echo "<p>{$alumnos[$i]}</p>";  
}
```

3.3. Matrices.

Indexada numéricamente

Se puede usar la función `array()` o la sintaxis `[]` para crear una matriz vacía antes de asignar los valores.

```
$alumnos = array();
```

```
$alumnos[0] = "Ana";
```

```
$alumnos[1] = "Juan";
```

```
$alumnos[2] = "Pedro";
```

3.3. Matrices.

Indexada numéricamente

Se puede crear la matriz asignándole valores directamente:

```
$alumnos = array ('Ana', 'Juan', 'Pedro');
```

También se pueden crear matrices con una secuencia ascendente de números o caracteres, usando la función `range()`.

```
range(mixed $start, mixed $end, number $step = 1): array
```

```
$numeros = range(1, 20, 2);
```

```
$letras = range('a', 'g');
```

3.3. Matrices.

Indexada numéricamente

Se puede conocer el número de elementos del array usando la función `count()`, o su alias `sizeof()`.

`count(mixed $array_or_countable, int $mode = COUNT_NORMAL): int`

- El parámetro `mode` se puede establecer `COUNT_RECURSIVE`.

```
for ($i=0; $i<count($numeros); $i++) {  
    echo "<p>{$numeros[$i]}</p>";  
}
```

3.3. Matrices.

Indexada numéricamente

Se puede utilizar el bucle foreach, para recorrer matrices sin conocer el número de elementos.

```
foreach ($numeros as $valor) {  
    echo "<p>$valor</p>";  
}  
  
foreach ($numeros as $clave=>$valor) {  
    echo "<p>{$numeros[$clave]} = $valor</p>";  
}
```

3.3. Matrices.

Indexada numéricamente

Si se quiere modificar el contenido de una matriz dentro de un bucle foreach se tienen que pasar los elementos por referencia usando &.

```
$numeros = array (1, 2, 3, 4);  
foreach ($numeros as &$valor) {  
    $valor = $valor * 2;  
}  
print_r($numeros);
```

3.3. Matrices. Asociativa

Permite establecer un nombre para cada una de las posiciones del array, de manera que a la hora de acceder a un elemento se utilice ese nombre en lugar de la posición que ocupa.

```
$capitales['España'] = "Madrid";  
$capitales['Francia'] = "Paris";  
$capitales['Italia'] = "Roma";
```

Utilizando array():

```
$capitales = array("España"=>"Madrid",  
                  "Francia"=>"Paris",  
                  "Italia"=>"Roma");
```


3.3. Matrices. Asociativa

Para recorrer la matriz se puede utilizar el bucle foreach.

```
foreach ($capitales as $clave => $valor) {  
    echo "<p>$clave: $valor</p>";  
}
```

3.3. Matrices.

Indexada bidimensional

Para acceder a un elemento necesitaremos dos índices numéricos.

```
$alumnos = array(  
    array("Ana", "Sánchez", "DAW1", 949123123),  
    array("Javier", "Pérez", "ASIR2", 949321321),  
    array("Carlos", "Sanz", "SMR1", 949222333)  
);
```

Para acceder al curso (columna 2) del alumno Javier (fila 1) habrá que usar `$alumnos[1][2]`.

3.3. Matrices.

Bidimensional asociativa

Representa una tabla que tiene cabeceras. Se necesitan 2 índices asociativos.

	Azuqueca	Jadraque	Horsche
Azuqueca	-	2-0	2-1
Jadraque	0-3	-	2-2
Horsche	0-1	1-0	-

3.3. Matrices.

Asociativa bidimensional

```
$azu = ["Azuqueca" => "-", "Jadraque" => "2-0",  
        "Horche" => "2-1"];  
$jad = ["Azuqueca" => "0-3", "Jadraque" => "-",  
        "Horche" => "2-2"];  
$hor = ["Azuqueca" => "0-1", "Jadraque" => "1-0",  
        "Horche" => "-"];  
$resultados = ["Azuqueca" => $azu,  
                "Jadraque" => $jad,  
                "Horche" => $hor  
];  
$res = $resultados['Azuqueca']['Horche'];
```

3.3. Matrices.

Asociativa bidimensional

```
$azu = array("Azuqueca" => "-", "Jadraque" => "2-0",  
            "Horche" => "2-1");  
$jad = array("Azuqueca" => "0-3", "Jadraque" => "-",  
            "Horche" => "2-2");  
$hor = array("Azuqueca" => "0-1", "Jadraque" => "1-0",  
            "Horche" => "-");  
$resultados = array("Azuqueca" => $azu,  
                    "Jadraque" => $jad,  
                    "Horche" => $hor  
);  
$res = $resultados['Azuqueca']['Horche'];
```

3.3. Matrices. Tridimensional

No tiene una representación natural inmediata.

```
$alumnos = array (  
    array (  
        array("Ana", "Sánchez", 949123123),  
        array("Javier", "Pérez", 949321321)  
    ),  
    array (  
        array("Juan", "Martínez", 949321123),  
        array("Maria", "Gómez", 949123321)  
    ),  
    array (  
        array("Abel", "Sanz", 949111222),  
        array("Laura", "Polo", 949333444)  
    )  
);
```

3.3. Matrices.

Tridimensional

```
$alumnos = [  
    [  
        ["Ana", "Sánchez", 949123123],  
        ["Javier", "Pérez", 949321321]  
    ],  
    [  
        ["Juan", "Martínez", 949321123],  
        ["Maria", "Gómez", 949123321]  
    ],  
    [  
        ["Abel", "Sanz", 949111222],  
        ["Laura", "Polo", 949333444]  
    ]  
];
```

3.3. Matrices. Operadores

Unión (+). La unión de dos matrices $a + b$ añadirá los elementos de la matriz b al final de a , pero *evitando las colisiones*.

- Si algún elemento de b posee una clave ya existente en a , ese elemento no se añade.

Copia (=). Hace una copia exacta de la matriz.

Igualdad (==). Compara dos matrices y devuelve true si ambas contienen las mismas parejas clave-valor.

3.3. Matrices. Operadores

Identidad (===). Compara dos matrices y devuelve `true` solo si ambas contienen las mismas parejas clave-valor, en el mismo orden y del mismo tipo.

Desigualdad (!= o <>). Compara dos matrices y devuelve `true` si no son iguales.

No identidad (!==). Compara dos matrices y devuelve `true` si no son idénticas.

3.3. Matrices.

Funciones. Ordenación

`sort(array &$array, int $flags = SORT_REGULAR): true`

Ordena la matriz en orden ascendente (números) o alfabético (cadenas).

- `flags`: especifica el tipo de ordenación que se desea: `SORT_REGULAR` (predeterminada), `SORT_NUMERIC`, `SORT_STRING`, `SORT_LOCALE_STRING`, `SORT_NATURAL`, `SORT_FLAG_CASE`.

```
$alumnos = array ("Juan", "Ana", "Pedro");
```

```
sort ($alumnos);
```

3.3. Matrices.

Funciones. Ordenación

`ksort(array &$array, int $flags = SORT_REGULAR): true`

`asort(array &$array, int $flags = SORT_REGULAR): true`

El proceso de ordenación en matrices asociativas exige que se mantengan los pares clave-valor

Habrà que elegir si se quiere ordenar por clave con `ksort()` o por valor con `asort()`.

3.3. Matrices.

Funciones. Ordenación

```
rsort(array &$array, int $flags = SORT_REGULAR): true  
arsort(array &$array, int $flags = SORT_REGULAR): true  
krsort(array &$array, int $flags = SORT_REGULAR): true
```

Las tres funciones son análogas a las funciones de ordenación, pero en orden descendente.

3.3. Matrices.

Funciones. Ordenación multidimensional

```
usort(array &$array, callable $callback): true  
uksort(array &$array, callable $callback): true  
uasort(array &$array, callable $callback): true
```

Las matrices multidimensionales no se pueden ordenar directamente.

- PHP no puede ordenar un elemento que es a su vez otra matriz.

Para ordenar estas matrices, el usuario tiene que definir una función de comparación que será llamada posteriormente.

- El parámetro `callback` almacenará el nombre de esta función.

En matrices asociativas, `uksort()` ordenará por claves y `uasort()` por valores.

3.3. Matrices.

Funciones. Ordenación multidimensional

```
$frutas[0]["fruta"] = "uvas";
$frutas[1]["fruta"] = "limones";
$frutas[2]["fruta"] = "manzanas";
function cmp($a, $b){
    return strcmp($a["fruta"], $b["fruta"]);
}
usort($frutas, "cmp");
echo '<pre>';
foreach ($frutas as $clave => $valor) {
    echo "\$frutas[$clave]: " . $valor["fruta"] . "\n";
}
echo '</pre>';
```

3.3. Matrices.

Funciones. Ordenación multidimensional

```
array_multisort(array &$array1, mixed $array1_sort_order = SORT_ASC,  
mixed $array1_sort_flags = SORT_REGULAR, mixed ...$rest): bool
```

Puede ordenar varias matrices a la o una matriz multidimensional por una o varias dimensiones.

El orden puede ser

- SORT_ASC: ascendente. Por defecto
- SORT_DESC: descendente.

3.3. Matrices.

Funciones. Ordenación multidimensional

El tipo puede ser

- `SORT_REGULAR`: comparación normal, sin cambiar tipos.
- `SORT_NUMERIC`: compara los elementos numéricamente.
- `SORT_STRING`: compara los elementos como si fueran cadenas.
- `SORT_LOCALE_STRING`: compara los elementos como si fueran cadenas, usando el `locale` actual, que se puede cambiar con `setlocale()`.
- `SORT_NATURAL` : compara los elementos por orden natural, como `natsort()`.
- `SORT_FLAG_CASE`: se puede combinar con `SORT_STRING` o `SORT_NATURAL` para que las comparaciones no sean sensibles a mayúsculas
 - Hay que usar el operador OR de bit.

3.3. Matrices.

Funciones. Ordenación multidimensional

```
$ar = array(  
    array("10", 11, 100, 100, "a"),  
    array( 1, 2, "2", 3, 1)  
);  
array_multisort($ar[0], SORT_ASC, SORT_STRING,  
                $ar[1], SORT_DESC, SORT_NUMERIC);  
var_dump($ar);
```

3.3. Matrices.

Funciones. Desplazamiento

Existe un puntero interno que apunta al elemento actual de la matriz.

`current(array|object $array): mixed`

- Devuelve el elemento actual. Tiene como alias `pos()`.

`next(array|object &$array): mixed`

- Avanza una posición y devuelve el elemento.

`prev(array|object &$array): mixed`

- Retrocede una posición y devuelve el elemento.

`reset(array|object &$array): mixed`

- Retrocede al primer elemento y devuelve su valor.

`end(array|object &$array): mixed`

- Avanza hasta el último elemento y devuelve su valor.

3.3. Matrices. Funciones. Otras

`shuffle(array &$array): true`

Mezcla la matriz: cambiar aleatoriamente el orden de los elementos.

- No se debe usar esta función para criptografía.

`array_reverse(array $array, bool $preserve_keys = false): array`

Devuelve una matriz igual que la original, pero con los valores en orden inverso.

3.3. Matrices.

Funciones. Otras

`array_walk(array|object &$array, callable $callback, mixed $arg = null): true`

Aplica una misma función sobre todos los elementos de la matriz.

- Si se quiere modificar el valor hay que pasarlo por referencia.
- Se pueden usar funciones anónimas.

```
$matriz = array (1, 2, 3, 4);  
function impr(&$dato){echo "<p>$dato</p>";$dato++;}  
array_walk ($matriz, 'impr');  
print_r($matriz);
```

3.3. Matrices. Funciones. Otras

`in_array(mixed $needle, array $haystack, bool $strict = false): bool`

Busca una aguja `$needle` en una matriz pajar `$haystack`.

`array_count_values(array $array): array`

Devuelve una matriz bidimensional que contiene los valores de la original como claves y su frecuencia (número de apariciones) como valor.

Actividad 3.10

- a. Dada la siguiente tabla con distancias entre ciudades españolas, crea una aplicación que te permita seleccionar la ciudad de origen y la de destino y te muestre en pantalla la distancia entre ambas Utiliza una matriz bidimensional indexada.
- b. Repite el ejercicio utilizando una matriz bidimensional asociativa.

	Barcelona	Coruña	Madrid	Sevilla
Barcelona	0	1188	621	1046
Coruña	1188	0	609	947
Madrid	621	609	0	538
Sevilla	1046	947	538	0

Actividad 3.11.

Cálculo de valores estadísticos

- a. Realiza una aplicación que permita calcular las funciones estadísticas básicas (mínimo, máximo, suma y media) sobre un conjunto de valores.
 - Para ello primero pide al usuario que introduzca el número de valores que tiene la muestra y la función a calcular.
 - A continuación solicita que introduzca los valores de la muestra a través de otro formulario, éste ya con tantos elementos para recoger valores como se hayan indicado anteriormente.
 - Por último realiza el cálculo indicado sobre los valores y muéstralo.
- b. Copia la aplicación y modifícala usando una matriz para almacenar los elementos.

Actividad 3.12

Crea un formulario que permita introducir información acerca de los artículos suministrados a la venta a través de la web (máximo 10 artículos):

- nombre del artículo
- su precio
- cómo desea ordenar los artículos: por nombre o por precio, y en orden ascendente o descendente.

A continuación, ordena los artículos siguiendo el criterio establecido y muéstralos en una tabla.

Actividad 3.13.

Juego de los trileros

Para comenzar muestra en pantalla

- Tres cartas en orden aleatorio, por ejemplo, as, dos y tres.
- Un botón para barajar las cartas.

Cuando el jugador pulse este botón las cartas serán barajadas y mostradas boca a abajo en pantalla para que el jugador trate de averiguar dónde se encuentra el as.

Actividad 3.14

- a. Crea un formulario para introducir información sobre máximo 10 alumnos de un centro: nombre, apellidos, curso, edad y localidad.
 - Ordena a los alumnos en orden alfabético por nombre.
- b. Ordena a los alumnos en orden alfabético por sus apellidos.
 - Si hay son iguales los apellidos se utilizará el nombre para desempatar.
- c. Usa la función `array_multisort` para realizar la aplicación anterior ordenando los datos de forma ascendente.
 - Utiliza los siguientes criterios: Curso, Apellidos y Nombre.
- d. Muestra la información en dos tablas, de principio a fin y de fin a principio, utilizando funciones de desplazamiento sobre matrices.

Actividad 3.15

- a. Crea un formulario que permita recoger 5 números de 1 a 100, almacénalos en una matriz.
- b. Modifica la matriz usando la función `array_walk` para que cada elemento sea la mitad de su valor original.

3.4. Cadenas de caracteres

Son series de caracteres, donde cada carácter ocupa un byte

- PHP solo admite un conjunto de hasta 256 caracteres.
 - Una cadena puede llegar a alcanzar hasta 2 GB (2147483647 bytes) de tamaño.
- Una cadena será codificada según la codificación del archivo del código.

No existe ninguna limitación sobre los valores que pueden componer una cadena.

Las extensiones `intl` y `mbstring` contienen funciones útiles para escribir correctamente aplicaciones web con soporte Unicode.

3.4. Cadenas de caracteres

La forma más sencilla de imprimir una cadena de caracteres es usarla construcción del lenguaje `echo`.

- No requiere el uso de paréntesis.
- No siempre se puede usar en el contexto de uso de una función.
- No añade nueva línea al final.
- Posee una sintaxis abreviada o *shorthand*.
 - `<?=$foo?>` equivale a `<?php echo $foo?>`

Otra construcción similar es `print`, que sí admite paréntesis.

- La diferencia principal es que `echo` admite varias cadenas separadas por comas, mientras que `print` solo admite una.

3.4. Cadenas de caracteres

Los literales de cadenas de caracteres se pueden especificar de cuatro formas diferentes:

- Entrecomillado simple
- Entrecomillado doble
- Sintaxis heredoc
- Sintaxis nowdoc

3.4. Cadenas de caracteres. Entrecomillado simple

Se muestra texto de forma literal

Solo admite el escape de un único carácter: la propia comilla simple

- Para escapar se usa siempre la barra invertida \ '.

```
echo 'Arnold dijo una vez : "I\'ll be back";
```

3.4. Cadenas de caracteres. Entrecomillado doble

Existen varias secuencias de escape. Entre otras:

- `\n` avance de línea
- `\r` retorno de carro
- `\t` tabulador horizontal
- `\\` barra invertida
- `\$` signo de dólar
- `\"` comillas dobles

<https://www.php.net/manual/es/language.types.string.php#language.types.string.syntax.double>

3.4. Cadenas de caracteres.

Sintaxis heredoc

Utiliza el operador <<< .

Tras el operador se debe escribir un identificador y una línea nueva.

- Este identificador debe seguir las reglas de nomenclatura de etiquetas de PHP.
- A continuación, debe escribirse la cadena de texto.
- Para cerrar se debe escribir el identificador escogido en la primera columna de la nueva línea, y terminar la instrucción PHP (;).

```
echo <<< EOT
```

```
Ejemplo de una cadena  
expandida en varias líneas  
empleando la sintaxis heredoc.  
EOT;
```

3.4. Cadenas de caracteres.

Sintaxis nowdoc

Igual que la sintaxis heredoc, pero el identificador inicial debe escribirse entre comillas simples.

```
$str = <<< 'EOT'
```

Ejemplo de una cadena
expandida en varias líneas
empleando la sintaxis nowdoc.

```
EOT;
```

```
echo $str;
```

3.4. Cadenas de caracteres.

Variables dentro de cadenas

Al utilizar comillas dobles o sintaxis heredoc se analiza la cadena para buscar variables. Hay dos tipos de sintaxis: simple y compleja.

- En la sintaxis simple todo conjunto de caracteres seguido de \$ se interpretará como variable, sustituyendo su valor.

```
$fruta = "manzana";  
echo "Bebió zumo de $fruta.";
```

- En la sintaxis compleja se debe escapar todo entre llaves {} sin dejar espacios. Las expresiones complejas solo admiten este tipo.

```
echo "Bebió zumo de {$fruta}.";
```

3.4. Cadenas de caracteres.

Acceso a los caracteres

PHP implementa las cadenas de caracteres como una matriz de bytes. Por ello, para modificar un único carácter de una cadena se accede por índice con la notación entre corchetes [].

```
$str = 'Esto es una prueba.';  
echo $str[0];  
echo $str[20];
```

3.4. Cadenas de caracteres.

Operadores

Concatenación (.)

- Es más eficiente y por tanto más recomendable usar la notación "" con sustitución de variables mejor la concatenación por la mejora en la eficiencia.

```
$a="Hola " . $var;  
$a="Hola $var";
```

Concatenación y asignación (.=)

Comparación (==)

- Comparación estricta (===)

<https://www.php.net/manual/es/types.comparisons.php>

3.4. Cadenas de caracteres.

Funciones. Eliminación de espacios

```
trim(string $string, string $characters = " \n\r\t\v\x00" ): string  
ltrim(string $string, string $characters= " \n\r\t\v\x00" ): string  
rtrim(string $string, string $characters= " \n\r\t\v\x00" ): string
```

Se pueden eliminar espacios u otros caracteres al inicio de la cadena, al final o en ambos.

- El segundo parámetro, opcional, es el listado de caracteres a eliminar.

```
$texto = "  lalala.";
echo '<p>'.trim($texto).'</p>';
echo '<p>'.rtrim($texto, ".").'</p>';
```

3.4. Cadenas de caracteres.

Funciones. Escape de caracteres

```
addslashes(string $string): string
```

```
stripslashes(string $string): string
```

A la hora de almacenar datos en una base de datos, hay caracteres que pueden causar problemas porque pueden ser interpretados como caracteres de control

- comillas (simples y dobles), barras invertidas y el byte NUL.

Para evitarlo, esos caracteres especiales deben ser escapados añadiendo una barra invertida delante de ellos.

```
$campo=addslashes(trim("O'Brien  "));
```

3.4. Cadenas de caracteres.

Funciones . Saltos de línea HTML

```
n12br(string $string, bool $use_xhtml = true): string
```

Inserta saltos de línea HTML antes de todas las nuevas líneas (`\r\n`, `\n\r`, `\n` y `\r`) de una cadena de caracteres.

El segundo parámetro obliga a utilizar saltos XHTML (`
`).

```
$texto = "Una linea\nY otra línea";  
echo n12br($texto);
```


3.4. Cadenas de caracteres. Funciones. Impresión

`printf(string $format, mixed ...$values): int`

`sprintf(string $format, mixed ...$values): string`

Permiten aplicar un formato a una cadena.

- `printf()` imprime la cadena resultante y devuelve la longitud de la cadena impresa
- `sprintf()` devuelve la propia cadena, que habrá que asignar a una variable.

```
$total = 25;
```

```
printf ("Importe Total: %5.2f", $total);
```

3.4. Cadenas de caracteres. Funciones. Impresión

El especificador de formato tiene esta forma:

`% ['carácter_relleno'] [-] [anchura] [.precision] tipo`

- `[carácter_relleno]`: carácter que se usará para rellenar espacio hasta la anchura establecida.
- `[-]`: rellena a la derecha en lugar de a la izquierda.
- `[anchura]`: tamaño mínimo que ocupará la variable.
- `[.precisión]`: número de posiciones decimales

3.4. Cadenas de caracteres. Funciones. Impresión

El especificador de formato tiene esta forma (sigue):

`% ['carácter_relleno'] [-] [anchura] [.precision] tipo`

- El tipo de la variable puede ser
 - b: entero representado como binario.
 - c: entero representado como carácter ASCII.
 - d: entero representado como decimal (con signo).
 - e, E: notación científica.
 - f, F: *float* representado en coma flotante.
 - g, G: equivalente a juntar %e o %E con %f.
 - o: entero representado en octal.
 - u: entero representado como decimal sin signo.
 - s: cadena de caracteres.
 - x, X: entero representado en hexadecimal.

3.4. Cadenas de caracteres.

Funciones. Mayúsculas y minúsculas

`strtoupper(string $string): string`

`strtolower(string $string): string`

Convierte una cadena a mayúsculas o a minúsculas.

`ucfirst(string $str): string`

`ucwords(string $string, string $separators = " \t\r\n\f\v"): string`

Pone en mayúsculas, si es alfabético el primer carácter de la cadena o la primera letra de cada palabra, respectivamente.

3.4. Cadenas de caracteres.

Funciones. Dividir y combinar cadenas

`explode(string $separator, string $string, int $limit = PHP_INT_MAX): array`

Devuelve una matriz de cadenas donde cada una es una subcadena de la original formada al dividirla aplicando el delimitador.

`implode(string $separator, array $array): string`

Une elementos de una matriz con un separador (opcional, por defecto es una cadena vacía).

- Hay una versión obsoleta eliminada en PHP 8
 - `implode(array $array, string $separator): string`

3.4. Cadenas de caracteres.

Funciones. *Tokenización* de cadenas

`strtok(string $string, string $token): string|false`

`strtok(string $token): string|false`

Extrae partes de una cadena de una en una utilizando el *token* dado.

- En la primera llamada habrá que especificar la cadena, pero en las sucesivas sólo es necesario indicar el *token*.

```
$palabra= strtok($texto, " ");  
while ($palabra!= "") {  
    echo $palabra."<br/>"; $palabra = strtok(" ");  
}
```

3.4. Cadenas de caracteres. Funciones. Subcadenas

`substr(string $string, int $offset, ?int $length = null): string`

Permite obtener una subcadena contenida en la cadena original `string`.

- `start`: indica la posición desde la que se quiere obtener la subcadena.
 - Si el valor es positivo se cuenta desde el principio de la cadena y si es negativo desde el final.
- `length`: si es positivo indica el número de caracteres que se quieren obtener desde inicio, si es negativo los caracteres que se dejan fuera contados desde el final de la cadena.
 - Si no se indica nada se toman todos los caracteres desde inicio hasta el final de la cadena.

```
$subcadena= substr($texto, 5, -10);
```

3.4. Cadenas de caracteres. Funciones. Longitud

`strlen(string $string): int`

Devuelve el número de caracteres que componen la cadena.

3.4. Cadenas de caracteres. Funciones. Comparación

`strcmp(string $string1, string $string2): int`

`strcasecmp(string $string1, string $string2): int`

Reciben dos cadenas y devuelve un número entero en función de su comparación alfabética.

- Si son iguales devuelve 0.
- Si `str1` es menor (va antes) que `str2` devuelve un valor negativo.
- Si `str1` es mayor (va después) que `str2` devuelve un valor positivo.
- `strcasecmp()` no discrimina entre mayúsculas y minúsculas.

```
if (strcmp($var1, $var2) != 0) { /*...*/ }
```

3.4. Cadenas de caracteres. Funciones. Comparación

`strnatcmp(string $string1, string $string2): int`

`strncasecmp(string $string1, string $string2, int $length): int`

Reciben dos cadenas y devuelven un número entero en función de su comparación **natural**, no alfabética. Útiles para números y para nombres de archivos.

- La cadena "2" es menor que "12".
- <https://github.com/sourcefrog/natsort>
- `strnatcasecmp()` no discrimina entre mayúsculas y minúsculas.

3.4. Cadenas de caracteres.

Funciones. Búsqueda

```
strstr(string $haystack, string $needle, bool $before_needle = false):  
string|false
```

Busca una cadena (`needle`, aguja) dentro de otra (`haystack`, pajar).

- Por defecto, devuelve `haystack` desde la primera ocurrencia de `needle` hasta el final.
- Si el tercer parámetro es `true` devuelve la parte anterior a la ocurrencia.
- `stristr()` no discrimina entre mayúsculas y minúsculas.

3.4. Cadenas de caracteres.

Funciones. Búsqueda

```
strrchr(string $haystack, string $needle, bool $before_needle = false):  
string|false
```

Busca una cadena (`needle`, aguja) dentro de otra (`haystack`, pajar).

- Por defecto, devuelve `haystack` desde la última ocurrencia de `needle` hasta el final.
- Si el tercer parámetro es `true` devuelve la parte anterior a la ocurrencia.
- Existe `strchr()` como alias de `strstr()`.

3.4. Cadenas de caracteres.

Funciones. Búsqueda de posición

`strpos(string $haystack, mixed $needle, int $offset = 0): int|false`

`strrpos(string $haystack, mixed $needle, int $offset = 0): int|false`

Devuelven la posición numérica de `needle`, no la subcadena.

- `strpos()` devuelve la primera instancia de `needle`.
- `strrpos()` devuelve la última instancia.
- El parámetro opcional `offset` permite indicar la posición de `haystack` a partir de la cual empezar la búsqueda.
 - Si es negativo, se buscará desde el final hacia atrás.

3.4. Cadenas de caracteres.

Funciones. Búsqueda de posición

`strpos()` y `strrpos()` devolverán `false` si no encuentran `needle` en `haystack`.

- Para distinguir ese `false` de la posición 0 (inicio de `haystack`) hay que utilizar el comparador de identidad (`===`).

```
$cad = "Hola mundo";  
$pos = strpos ($cad, "H");  
if ($pos === false) {  
    echo "No encontrado";  
} else {  
    echo "Encontrado: $pos";  
}
```

3.4. Cadenas de caracteres.

Funciones. Reemplazo

```
str_replace(array|string $search, array|string $replace, string|array  
$subject, int &$count = null ): string|array
```

Sustituye todas las instancias de `search` en `subject` por `replace`, y devuelve la cadena sustituida. Sirve para corregir ortografía, traducir términos, etc.

- El parámetro opcional `count` guarda el número de sustituciones realizadas.
- `str_ireplace()` no discrimina entre mayúsculas y minúsculas.

3.4. Cadenas de caracteres.

Funciones. Reemplazo

```
substr_replace(array|string $string, array|string $replace, array|int  
$offset, array|int|null $length = null ): string|array
```

Copia y devuelve `string` sustituyendo la parte definida desde `start` (y `length` si lo hubiere) con el contenido de `replace`.

- Si el valor de `offset` es negativo se empieza a contar desde el final.
- Si `length` es positivo indica el número de caracteres a sustituir, si es 0 se inserta `replacement` sin borrar caracteres, y si es negativo indica el número de caracteres desde el final donde dejará de sustituir.

3.4. Cadenas de caracteres.

Expresiones regulares

Una expresión regular está formada por caracteres estándar y un conjunto de expresiones comodín que tendrán un significado concreto.

Las expresiones regulares se usan cuando interesa hacer una búsqueda en la que la cadena a buscar siga un determinado patrón.

Una vez creada la expresión regular, se puede utilizar para comprobar la validez de una cadena, encontrar el patrón en la subcadena, dividir la cadena según el patrón, etc.

3.4. Cadenas de caracteres.

Expresiones regulares. Funciones.

En general las funciones de expresiones regulares son menos eficientes que las correspondientes de cadenas.

Referencias:

- <https://www.php.net/manual/es/reference.pcre.pattern.syntax.php>
- <https://regex101.com/>
- <https://regexpr.com/>

3.4. Cadenas de caracteres.

Expresiones regulares. Caracteres básicos

Expresión	Tipo	Descripción
.	Carácter	Cualquier carácter individual excepto \n
[aeiou]	Valores	Cualquier carácter de los indicados
[a-z]	Rango	Cualquier carácter del rango incluidos
[a-zA-z]	Conjunto de rangos	Cualquier carácter de cualquier rango
[^a]	Exclusión	Cualquier carácter excepto el indicado
[^a-<]	Exclusión de rango	Cualquier carácter excepto caracteres del rango indicado

3.4. Cadenas de caracteres.

Expresiones regulares. Clases carácter

Clase	Descripción
<code>[:alnum:]</code>	Caracteres alfanuméricos
<code>[:alpha:]</code>	Caracteres alfabéticos
<code>[:lower:]</code> / <code>[:upper:]</code>	Letras minúsculas / mayúsculas
<code>[:digits:]</code> / <code>[:xdigits:]</code>	Dígitos decimales (\d) / hexadecimales
<code>[:space:]</code>	Caracteres de espaciado, incluyendo VT (casi \s)
<code>[:blank:]</code>	Tabuladores y espacio
<code>[:print:]</code>	Caracteres imprimibles incluyendo el espacio
<code>[:graph:]</code>	Caracteres imprimibles, excluyendo el espacio
<code>[:punct:]</code>	Caracteres imprimibles, excluyendo letras y dígitos
<code>[:cntrl:]</code>	Caracteres de control
<code>[:word:]</code>	Caracteres de palabra (\w)

3.4. Cadenas de caracteres.

Expresiones regulares. Repetición

Carácter	Descripción
*	El patrón se repite 0 o más veces.
+	El patrón se repite 1 o más veces.
?	El patrón se repite 0 o 1 vez.

3.4. Cadenas de caracteres.

Expresiones regulares. Metacaracteres

Metacarácter	Descripción	Equivalente
\b	Límite de palabra, al inicio o al final.	
\d	Cualquier carácter de dígito	[0-9]
\d	Cualquier carácter que no sea un dígito	[^0-9]
\s	Cualquier carácter de espaciado (espacios, tabulaciones, saltos de página o saltos de línea)	
\S	Cualquier carácter que no sea de espaciado	
\w	Cualquier carácter alfanumérico o guion bajo	[a-zA-Z0-9_]
\W	Cualquier carácter que no sea alfanumérico ni guion bajo	[^a-zA-Z0-9_]

3.4. Cadenas de caracteres.

Expresiones regulares. Metacaracteres

Metacarácter	Descripción
	Todo elemento separado por cumple el patrón.
^	El patrón debe de estar al inicio de la cadena.
\$	El patrón debe de estar al final de la cadena.

3.4. Cadenas de caracteres.

Expresiones regulares. Metacaracteres

Rama: |

Todo elemento separado por | es una opción que cumple el patrón.

- 1|2|3: uno de los tres caracteres
- com|net|edu: una de las tres cadenas

Anclajes: ^ y \$

^(file): la cadena debe comenzar con "file".

[a-z]\$: la cadena debe finalizar con una letra (a-z).

3.4. Cadenas de caracteres.

Expresiones regulares. Subexpresiones

Permiten dividir la expresión en partes.

Se utilizan paréntesis ().

Se puede especificar el número de repeticiones con expresiones numéricas entre llaves {}.

(muy)*grande: cualquier número de "muy " seguido de "grande".

- "grande" (0 muy), "muy grande", "muy muy grande", ...

(muy){2} grande : exactamente 2 repeticiones de muy.

(muy){1,3} grande : entre una y tres repeticiones de muy.

(muy){2,} grande: al menos 2 repeticiones de muy.

3.4. Cadenas de caracteres.

Expresiones regulares. Especiales

Hay que tener cuidadoso a la hora de buscar coincidencias con caracteres especiales

- `.,\|()[]{}^$-+*`.
- <https://www.php.net/manual/de/regexp.reference.meta.php>

Para buscar uno de esos caracteres específicamente habrá que utilizar la barra invertida (`\`) para escaparlo.

- Algunos de esos caracteres como `\` y `$` ya tienen un significado especial en PHP, por lo que para evitar que se empleen con el significado que tienen en PHP habrá que duplicar las barras: `\\` y `\\$`.

3.4. Cadenas de caracteres.

Expresiones regulares. Construcción

En PHP, para crear un patrón, se construye la expresión regular y se encierra entre el carácter delimitador barra (/).

```
$patron = '/php/';
```

Si después del delimitador final se añade el carácter `i` no se distingue entre mayúsculas y minúsculas en todo el contenido de la expresión.

```
$patron = '/php/i';
```

3.4. Cadenas de caracteres.

Expresiones regulares. Búsqueda

```
preg_match(string $pattern, string $subject, array &$matches = ?, int $flags = 0, int $offset = 0): int|false
```

Busca elementos de `subject` que coinciden con el patrón `pattern`.

Devuelve 1 si hay coincidencias, 0 si no hay y `false` si hay un error.

- `matches`: parámetro opcional que almacena los elementos que coincidan con el patrón.
- `flags`: admite el valor `PREG_OFFSET_CAPTURE`, para indicar que en la matriz se almacene, además de los elementos coincidentes, su posición.
- `offset`: se usa para indicar desde que posición de `subject` se quiere empezar a buscar.

3.4. Cadenas de caracteres.

Expresiones regulares. Búsqueda

`preg_match`

```
$sujeto = "abcdef";  
$patron = '/def/';  
preg_match($patron, $sujeto, $coinciden, PREG_OFFSET_CAPTURE);  
print_r($coinciden);
```

3.4. Cadenas de caracteres.

Expresiones regulares. Sustitución

```
preg_replace(string|array $pattern, string|array $replacement,  
string|array $subject, int $limit = -1, int &$count = null ):  
string|array|null
```

Busca los elementos de `subject` que coinciden con `pattern` y los sustituye por `replacement`.

- `limit`: número máximo de sustituciones, por defecto es ilimitado.
- `count`: permite almacenar el número de sustituciones realizadas.

3.4. Cadenas de caracteres. Expresiones regulares. Sustitución

preg_replace

```
$cad = 'Septiembre 15, 2017';  
$patron = '/septiembre/i';  
$sust = 'Octubre';  
echo preg_replace($patron, $sust, $cad);
```

3.4. Cadenas de caracteres.

Expresiones regulares. División

`preg_split(string $pattern, string $subject, int $limit = -1, int $flags = 0): array`

Divide la cadena `subject` utilizando el `pattern` indicado:

- `limit`: número máximo de elementos a obtener, por defecto es ilimitado.
- `flags`: opciones de sustitución.
- <https://www.php.net/manual/es/function.preg-split.php>

```
$claves = preg_split("/[[:blank:]],+/",  
    "hypertext language, programming");  
print_r($claves);
```


Actividad 3.16

- a. Recoge una cadena de texto a través de un formulario de manera que contenga espacios en blanco al principio y al final u otros caracteres repetidos. Muestra en pantalla la salida tras aplicar las funciones de limpiar cadenas.
- b. Crea un traductor de castellano a jerigonza inglesa *pig latin*, que recoja un texto (utiliza `textarea`), y lo muestre en pantalla "traducido". Para traducir a *pig latin* sólo tienes que tomar cada palabra del castellano
 - si empieza por vocal se añade al final "ay"
 - si empieza por consonante se pasa el grupo de consonantes del principio al final y después se añade ay.

Actividad 3.17

Crea una aplicación que permita recoger un texto (SMS) y lo comprima lo máximo posible antes de ser enviado. Para ello se sustituirán determinadas palabras por sus abreviaturas:

Palabra	Abreviatura
adiós	a2
besos	bs
dónde	dnd
fin de semana	finde
instituto	ies
mensaje	msj
por favor	pls
porque	xq
que	q
también	tb
por	x
para	xa

Actividad 3.18

Buscar y reemplazar

- Crea una pequeña aplicación que permita introducir un texto y nos permita buscar y/o reemplazar palabras dentro del texto. Habrá dos cajas de texto: "buscar" y "reemplazar", y un botón "Enviar".
- Al pulsar el botón
 - si hay contenido tanto en buscar como en reemplazar, se sustituirá cada ocurrencia de "buscar" dentro del texto por "reemplazar".
 - si sólo hay contenido en "buscar", se resaltarán todas las ocurrencias de éste en el texto.

Actividad 3.19

Crea un formulario para recibir sugerencias de usuarios que conste de los siguientes elementos:

- nombre, apellidos y correo electrónico del usuario
- departamento al que se dirige el mensaje (facturación, ventas, servicio técnico)
- tema
- mensaje

Además de estos datos se tomará la fecha del sistema.

Recoge los datos del formulario y genera un mensaje de respuesta automático utilizando la siguiente plantilla que tenemos almacenada en una variable de tipo `string`.

- El mail de cada departamento es el nombre del departamento seguido de "@daw2.com".

DE:	<<Mail_Departamento>>
PARA:	<<Mail_Usuario>>
TEMA:	<<Tema>>
MENSAJE:	
Estimado Sr. <<Apellidos>>.	
En relación a su mensaje del día <<Fecha>>, le informamos que nuestro personal de <<Departamento>> está realizando las gestiones necesarias para darle la mejor respuesta.	
Reciba un cordial saludo.	

Actividad 3.20

Dada una cadena \$cad utiliza expresiones regulares para comprobar si contiene los siguientes elementos:

- a. Un número DNI correcto.
- b. Un número de teléfono fijo o móvil.
- c. Un nombre de un archivo válido con extensión php, css, html, htm.
- d. Una fecha (dd/mm/aaaa ó dd-mm-aaaa).
- e. Una dirección IPv4.
- f. Una dirección de correo electrónico válida con dominio .com, .org o .es

Actividad 3.21

- a. La variable global `$_SERVER` es una matriz que contiene información como cabeceras, rutas de acceso, etc. del servidor web. Realiza un programa que muestre cada uno de los campos de esta matriz y su correspondiente valor ocultando las rutas de archivos y direcciones de correo que serán sustituidas por `<<RUTA>>` y `<<EMAIL>>` respectivamente.
- b. Crea una aplicación web que permita generar una plantilla a partir de un texto. Para ello la aplicación recogerá un texto introducido por el usuario y a continuación buscará en el texto elementos que identifique como: DNIs, fechas, teléfonos y direcciones de email, y sustituirá cada ocurrencia de estos elementos por un código de campo que vendrá dado por el tipo de elemento encerrado entre `<<>>`, por ejemplo: `<<fecha>>`, `<<dni>>`.

Actividad 3.22.

Pizzería

Crea una aplicación web para una pizzería que permita realizar pedidos online.

- El formulario de pedido deberá recoger las siguientes opciones:
- Tamaño de la pizza (elegir una): mini (2.95€), media (4.95€) y maxi (8.95€) (base normal).
- Base (elegir una): normal (0€), crujiente (1€) y rellena (2€).
- Salsa (se puede elegir una o ninguna): barbacoa (0,95€), carbonara (1,45€).
- Ingredientes (tantos como se deseen): pollo (0,55€), bacon (0.75€), jamón (0.95€), cebolla (0.45€), aceitunas (0.55€), pimienta (0.65€).
- Una vez recibido el pedido se mostrará en pantalla una factura detallada

Todos los importes de la factura se tienen que mostrar en formato euro (2 decimales y el símbolo €), alineados a la derecha y rellenando con 0 a la izquierda de manera que la parte entera ocupe siempre 2 cifras.

3.5. Funciones definidas por el usuario

Una función es un fragmento o bloque de código autónomo realiza una operación. Define una interfaz de llamada, que puede incluir o no parámetros, y puede devolver como máximo un único resultado.

- En PHP existen numerosas funciones predefinidas para múltiples propósitos.

La verdadera potencia de las funciones reside en las funciones definidas por el usuario.

- Están disponibles en principio solo en el *script* que el que se definen.

PHP no soporta sobrecarga de funciones.

3.5. Funciones definidas por el usuario.

Definición

Una función definida por el usuario en PHP debe tener la siguiente sintaxis:

```
function foo($arg_1, $arg_2, /* ..., */ $arg_n) {  
    /* ..., */  
    return $retval;  
}
```

3.5. Funciones definidas por el usuario.

Definición. Nombre

foo

El nombre de la función debe ser corto y descriptivo.

- Existen ciertas limitaciones:
 - Sólo puede estar formado por letras, dígitos y guiones bajos.
 - No puede comenzar por dígito.
 - No se puede utilizar el mismo nombre asignado a otra función.
- A diferencia de los nombres de variables, PHP no discrimina mayúsculas y minúsculas en los nombres de las funciones
 - Las funciones `name()` y `Name()` son equivalentes.

3.5. Funciones definidas por el usuario.

Definición. Parámetros

`($arg_1, $arg_2, /* ..., */ $arg_n`

Permiten introducir variables en el contexto la función.

Para utilizar parámetros hay que establecer la lista de parámetros en la definición de la función, y pasar los valores adecuados cuando se realice la llamada.

```
function suma ($num1, $num2) {  
    $resultado = $num1 + $num2;  
    return $resultado;  
}  
echo suma (2,5);
```

3.5. Funciones definidas por el usuario.

Definición. Parámetros. Opcionales

Pueden estar o no en la llamada a la función.

Hay que incluirlos en la definición de la función asignándoles un valor por defecto.

```
function incremento($num1, $inc=1) {  
    return $num1 + $inc;  
}  
$result = incremento (2);  
echo incremento ($result, 3);  
echo incremento ($result, inc: 3);
```

3.5. Funciones definidas por el usuario.

Definición. Parámetros. Valor/referencia

Paso por valor

- La función crea una copia local de la variable, que es la que utiliza
- El valor de la variable original no se modifica.
- Es el comportamiento por defecto en el paso de parámetros.

Paso por referencia

- Se referencia la variable original. Las modificaciones se realizan sobre ella.
- Hay que indicarlo explícitamente con el símbolo & en la definición.

```
function nombre_funcion (&$param1, $param2) {  
    // Aquí se puede modificar solo $param1  
}
```

3.5. Funciones definidas por el usuario.

Definición. Parámetros. Por valor/referencia

Paso por referencia. Ejemplo.

```
function decremento(&$num, $inc) {  
    $num = $num - $inc;  
}  
  
$num = 5;  
  
decremento ($num, 3);  
  
echo "Valor de num: $num.";
```

3.5. Funciones definidas por el usuario.

Definición. Parámetros. Tipos

PHP no fuerza que los parámetros tengan un tipo, pero **se puede indicar un tipo en la definición para que eleve un error capturable si no coinciden los tipos al llamar a la función.**

- Es recomendable usarlo siempre
 - Sobre todo cuando el tipo del parámetro influya en el funcionamiento de la función.

La mayoría de tipos son válidos, incluyendo tipos unión en intersección.

- Hay que usar los nombres de tipos originales (`bool`, `int`, `float`, `string`), no sus alias.

```
function test(bool $param) {}
```

3.5. Funciones definidas por el usuario.

Definición. Parámetros. Tipos

Por defecto PHP no devolverá un error fatal si el tipo de la declaración no coincide con el del valor. Para ello se puede usar el constructo `declare()`.

```
declare(strict_types=1);
```

Esta declaración debe ser la primera de todas.

```
function sum(int $a, int $b) {  
    return $a + $b;  
}  
var_dump(sum(1.5, 2.5));  
//Argument #1 ($a) must be of type int, float given
```


3.5. Funciones definidas por el usuario.

Definición. Parámetros. Cantidad variable

`func_num_args(): int`

`func_get_args(): array`

`func_get_arg(int $position): mixed`

Estas tres funciones sirven para saber el número de parámetros pasados y obtener todos ellos (como una matriz) o uno solo de ellos.

3.5. Funciones definidas por el usuario.

Devolución de valores

Las funciones pueden devolver un valor, que se puede utilizar fuera de la función.

Para devolver un valor, hay que utilizar la instrucción `return` seguida del valor a devolver, que puede ser una expresión.

- Sin ningún valor ni expresión, `return` puede utilizarse para finalizar una función y devolver la ejecución al punto de llamada.

```
function resta ($num1, $num2) {  
    return $num1 - $num2;  
}  
$valor = resta (5,3);
```

3.5. Funciones definidas por el usuario. Devolución de valores

Desde PHP 7, una función también puede devolver otra función.

```
function foo () {  
    return function ($a){return 'Salida: '.$a;};  
}  
  
echo '<p>'.foo()('b').'</p>';  
  
$valor = foo();  
echo '<p>'.$valor('b').'</p>';
```

3.5. Funciones definidas por el usuario.

Devolución de valores. Tipo devuelto

También se puede especificar el tipo del valor que será devuelto por una función.

- Están disponibles los mismos tipos para las declaraciones de tipo de devolución que para las declaraciones de tipo de argumento.

En el modo predeterminado de tipificación débil, los valores devueltos serán forzados al tipo correcto si no son ya de ese tipo.

En el modo fuerte, el valor devuelto debe ser del tipo correcto, o de lo contrario se lanzará una excepción

- Si se añade ?, además se puede devolver null.

3.5. Funciones definidas por el usuario.

Devolución de valores. Tipo devuelto

Ejemplo

```
function subtract($a, $b): ?float {  
    // será devuelto un float o null  
    return $a - $b;  
}  
var_dump(subtract(1, 2));
```

3.5. Funciones definidas por el usuario.

Funciones anónimas

Se pueden definir funciones sin nombre.

- En un parámetro que admita el tipo callable.

```
echo preg_replace_callback('~-([a-z])~',  
    function ($coincidencia) {  
        return strtoupper($coincidencia[1]);  
    },  
    'hola-mundo');
```

3.5. Funciones definidas por el usuario.

Funciones anónimas

Se pueden definir funciones sin nombre.

- Asignándolo a una variable.

```
$saludo = function($nombre){  
    printf("Hola %s\r\n", $nombre);  
};  
$saludo('Mundo');
```

3.5. Funciones definidas por el usuario. Llamadas

Una función puede ser llamada desde cualquier lugar del *script*.

- No tienen que estar definidas previamente, de no ser que la definición sea condicional.

Para realizar la llamada, simplemente hay que poner el nombre de la función acompañada de los parámetros que la función espere recibir.

```
echo add (5,3);  
/* ... */  
function add ($num1, $num2) {  
    return $num1 + $num2;  
}
```


3.5. Funciones definidas por el usuario. Llamadas. Recursividad

PHP permite el uso de funciones recursivas, que son funciones que se llaman así mismas.

Es una solución matemáticamente elegante, pero desde el punto de vista computacional es más costosa que la versión iterativa de la misma función.

Actividad 3.23

Repita la aplicación de actividad 3.11 (*Cálculo de valores estadísticos*) utilizando funciones.

En este caso calcularemos: suma, media, máximo, mínimo, mediana, desviación típica y moda.

Actividad 3.24.


Pétalos alrededor de la rosa

Crea una aplicación que permita jugar a este juego **utilizando funciones**.

El juego consiste en lanzar y mostrar 5 dados y solicitar al usuario que adivine el número de pétalos que hay alrededor de la rosa.

- Los pétalos alrededor de la rosa son el número de puntos que hay en todos los dados alrededor del punto central de cada uno de ellos.

Petalos Alrededor De La Rosa



Nombre:

Pétalos: