Boletín de Ejercicios y Prácticas

PHP

Contenido

Notas

Los ficheros que contengan cada uno de los apartados/ejercicios deberán usar los nombres p1_X.php, sustituyéndose por el número de ejercicio. Por ejemplo, p1_4.php para el ejercicio 4 de la parte 1, salvo que el ejercicio indique lo contrario.

Se creará una carpeta para clasificar cada uno de los ejercicios/prácticas de las partes y otra carpeta para la práctica 3 y otra para la 4.

Ejercicios parte 1. Introducción

1. Script PHP que calcule y muestre la suma y el producto de los 10 primeros números naturales.

2. Script php que, dados tres números, los ordene de menor a mayor.

3. Script php que genere y muestre 10 números aleatorios (del 1 al 500) y diga cuál es el mayor.

4. Script php que calcule la suma de los 10 primeros números pares y el producto de los 10 primeros números impares, simultáneamente, y que muestre los resultados por pantalla.

5. Script php que muestre para los 15 primeros números naturales, el número, su doble y su triple.

Ejercicios parte 2. Conceptos avanzados

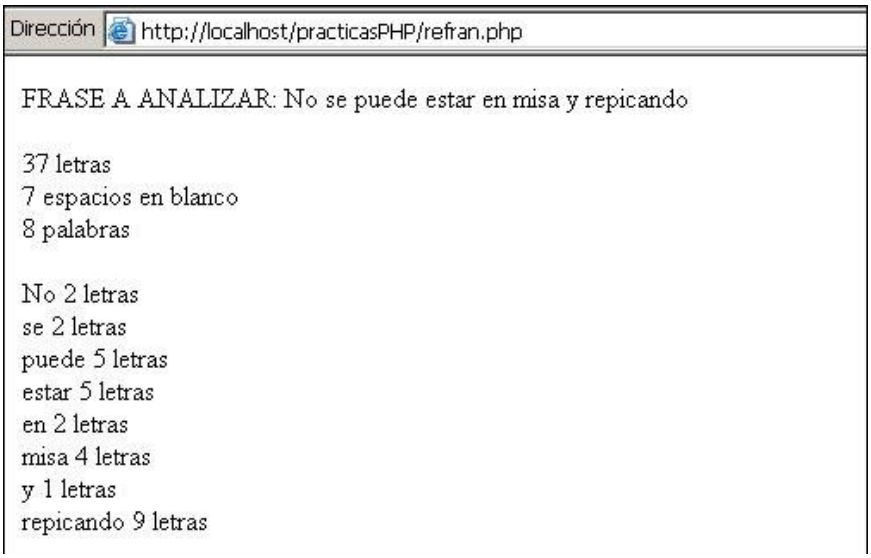
Práctica 1

Crea la función mostrar_impares que muestre los caracteres en posiciones impares de una cadena predefinida. Ejecútalo con la frase "A quien madruga Dios le ayuda".

Práctica 2

Muestra de la frase "El perro de San Roque no tiene rabo":

- Las letras totales de la frase.
- El número de palabras de la frase.
- Una línea con el número de letras de cada palabra.



Práctica 3

Comprobar si un NIF es válido. Un NIF ha de constar exactamente de 8 números y una letra. Para comprobar si un carácter es un número o una letra se puede usar la función ord() que nos da el código ASCII de dicho carácter. En el código ASCII los números se encuentran en las posiciones 48 a 57 y las letras en las posiciones 65 a 90 (mayúsculas) y 97 a 122 (minúsculas). Una vez comprobado que es correcto debe mostrar el DNI por pantalla con la última letra en mayúsculas, independientemente de como estuviera en el dato de entrada.

Un ejemplo con llamadas a la función ord():

```
<?php
echo ord("7") . "<br>"; // Vale 55, luego es un número
echo ord("c") . "<br>"; // Vale 99, luego es una letra
echo ord("-") . "<br>"; // Vale 45, ni letra ni número
?>
```

Práctica 4

Mejora el ejemplo 7 del tema para comprobar que la dirección de email no contiene espacios en blanco.

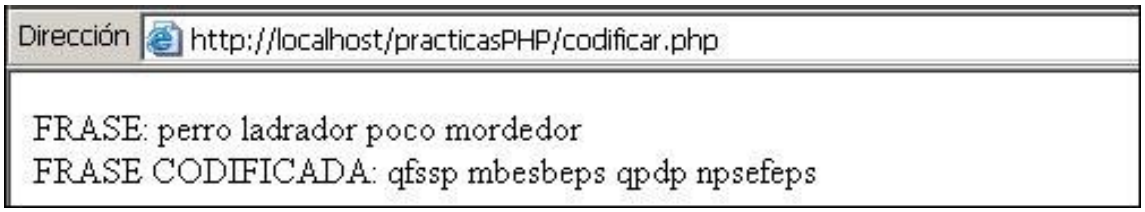
Práctica 5

Haz un array indexado numéricamente que contenga las letras del abecedario en minúsculas. Luego recórrelo, y muéstralo por pantalla cada letra del array, primero en minúsculas y luego en mayúsculas.

Práctica 6

Haz un programa para codificar por desplazamiento una frase utilizando el array de letras del ejercicio anterior. La idea es que convierta cada letra por la siguiente del abecedario y a la última le asigne la primera letra: a la "a" le corresponde la "b", a la "b" la "c", y así sucesivamente hasta que a la "z" la "a". Si la frase contiene espacios debe dejarlos igual.

Sugerencia: puede resultar útil la operación módulo para tratar los índices del array.



Práctica 7

Modifica el programa anterior para que el desplazamiento sea variable. Es decir, en un desplazamiento 3 se transforma la letra por la que está 3 posiciones más adelante.

Práctica 8

En esta práctica se pide crear una array asociativo con parejas de datos nombres de persona - altura. Luego se usará una estructura foreach para recorrerlo y mostrar, por cada elemento del array, el mensaje correspondiente del tipo "María mide 1.75 m".

Práctica 9

Vuelve a programar la práctica 5 del tema anterior pero esta vez usando arrays asociativos. Las estructuras de datos necesarias son:

- Un array $precio_kg. Cada elemento tiene como clave el nombre del producto y como valor el precio por kg del mismo.
- Un array $lista_compra. Cada elemento tiene como clave el nombre del producto y como valor la cantidad en kg comprada.

Una vez almacenados los datos en los arrays procésalos mediante bucles para obtener el mismo resultado que en el ejercicio 8.

Práctica 10

Escribe el código para las siguientes funciones.

- Función pesetas_a_euros($pesetas): recibe como parámetro un valor en pesetas y devuelve el equivalente en euros (1 euro = 166.368 ptas).
- Función euros_a_pesetas($euros): recibe como parámetro un valor en euros y devuelve el equivalente en pesetas.

Modifica el programa del ejercicio 9 para mostrar el tiquet de compras en euros y en pesetas haciendo la conversión con estas funciones.

Práctica 11

Divide el script anterior en dos. Uno que contenga las funciones (biblioteca de funciones), llamado “conv_precios.php”, y otro que lo utilice luego.

Práctica 12

Extrae el código del ejercicio 7 en una función codificar() que reciba como parámetros una cadena de texto y un número, y devuelva el texto codificado con ese desplazamiento. Incluye en dicho script otra función que decodificar() que haga la operación inversa. Luego guárdalo como una biblioteca de funciones con el nombre “cripto.php”.

Práctica 13

Declara en un script un array $ciudades cuyos elementos contengan nombres de ciudades. Haciendo uso de la biblioteca “cripto.php” creada en la práctica anterior codifica estas palabras y guárdalas en un nuevo array llamado $codificado.

Luego decodifica esta lista y guárdala en un array nuevo llamado $decodificado.

Finalmente recorre los tres arrays y muestra por pantalla cada palabra antes de codificarla, tras codificarla y después de volverla a decodificar.

Práctica 14

Hay que hacer un formulario “zodiaco.html” que disponga de un campo de selección única para elegir el signo del zodiaco. Estos datos los ha de recoger la página “horoscopo.php” que mostrará el signo escogido.

Práctica 15

Haz un formulario que solicite una frase y un desplazamiento con el nombre “codificar.html” y, utilizando la biblioteca de funciones “cripto.php” de la práctica 12 muestre el resultado de la frase codificada en la página “codificado.php”.

Práctica 16

Amplia el ejemplo X para que al jugar se guarde un registro de partidas ganadas, empatadas y perdidas. Truco: Como los contenidos de las variables se pierden cada vez que se recarga la página es necesario enviarlos cada vez, se puede ampliar el formulario con tres campos ocultos que almacenen esta información:

```
<input type="hidden" name="ganadas" value="<?= $_REQUEST["ganadas"] ?>">
<input type="hidden" name="empatadas" value="<?= $_REQUEST["empatadas"] ?>">
<input type="hidden" name="perdidas" value="<?= $_REQUEST["perdidas"] ?>">
```

Práctica 17

Hay que realizar un programa que consiste en un rudimentario formulario de acceso. Este ejercicio constará un total de 6 páginas.

- *usuarios.php*: es el siguiente script PHP, que contiene en un array los usuarios que están registrados por el sistema.

```
<?php
// Array que contiene parejas de datos usuario-contraseña
$usuarios = array("Juan" => "draco",
"Luisa" => "baobab",
"Antonio" => "olmo");
?>
```

- *accesos.txt*: registro de entradas.
- *login.html*: es un formulario que pide un nombre y una contraseña.
- *verificar.php*: es una página que comprueba que los datos introducidos estén en el archivo “usuarios.php”.
    - Si el usuario es reconocido escribe en el fichero accesos.txt la entrada: "El usuario $usuario ha accedido al sistema." y lo redirecciona a la página “ok.php”.
    - Si el usuario no es reconocido escribe en el fichero accesos.txt la entrada: "Intento fallido de acceso del usuario $usuario." y lo redirecciona a la página “error.html”.
- *ok.php*: Muestra un mensaje "Usuario verificado. Está en una zona privada.".
- *error.html*: Muestra un mensaje "Usuario o contraseña desconocido".

Práctica 18

Crea un script que liste los archivos de un directorio que esté por debajo de aquel en el que esté guardado el programa, como por ejemplo la subcarpeta “listar”. (Para probarlo habrá que crear el directorio y guardar en él algunos archivos de muestra).

Práctica 19

Mejorar el programa anterior para que liste solo los contenidos del directorio que sean de un solo tipo (por ejemplo con la extensión .php).

Práctica 20

Esta práctica pone en juego muchos de los conceptos adquiridos hasta ahora y tiene una dificultad notablemente más alta. Se recomienda ir haciéndola por partes y no pasar a la siguiente hasta que no se ha dejado la anterior bien hecha.

Consiste en hacer una web que funcione como una hemeroteca. Los usuarios podrán subir y descargar archivos de ella. Una implementación orientativa son las siguientes páginas:

- *index.php*: Página de bienvenida. Ofrece enlaces a “subir.php” y “listar.php”.
- *subir.php*: Página con un formulario para subir un archivo. Lleva a la página “confirmar.php”.
- *confirmar.php*: Página que muestra un mensaje indicando que el archivo ha sido subido. Ofrece enlaces a “subir.php” y “listar.php”.
- *listar.php*: Muestra una lista con enlaces a todos los archivos que se han subido. Ofrece un enlace a “subir.php”.

Los archivos subidos no deberán ser guardados en el mismo directorio de las páginas PHP sino en uno en un nivel inferior llamado "descargas".

Recomendaciones:

- Usar un array para guardar la lista de los archivos.
- Seguir las recomendaciones de separación de código:
  - Poner la mayor parte del código PHP al comienzo de la página web.
  - Indentar el código para mayor claridad.
- Seguir las siguientes recomendaciones a la hora de programar:
  - No intentar hacerlo todo a la vez, separar los problemas. Una vez resuelto uno pasar al siguiente.
  - Si salen muchas líneas de código es que algo no está bien. Estudiar si hay una solución más sencilla al mismo problema.
  - Cuando se encuentran fallos viene bien poner en puntos clave del código sentencias echo "
DEPURACIÓN: $variable
" para ver el valor que toman las variables durante la ejecución de la página.
  - Estas sentencias se borran o comentan cuando la página esté terminada.

- Si se quiere hacer alguna mejora sobre el programa base hacerlo cuando este ya funciona correctamente, y habiendo guardado una copia de seguridad por si algo saliera mal.

Se sugieren las siguientes mejoras sobre el programa base:

FÁCIL: Mostrar información adicional del archivo subida en la página “confirmar.php” (nombre, extensión, tamaño).

MEDIA: Mostrar columnas adicionales en la página “listar.php” con la información extra del archivo (extensión, tamaño).

MEDIA: Ordenar en la página “listar.php” los archivos por orden alfabético del nombre de archivo.

Práctica 3. Conceptos avanzados

Especificación

Este ejercicio consiste en hacer una página web dinámica que funcione como libro de visitas. Concretamente ha de permitir:

- Ver la lista de los comentarios de los visitantes.
- Introducir un comentario por parte de un visitante.

Esta práctica toma como base la práctica 17. Una implementación orientativa sería la siguiente:

- *index.php*: página de bienvenida con enlaces a “libro_visitas.php” y “nueva_visita.php”.
- *libro_visitas.php*: página que lista las visitas. Incluye también un enlace a “nueva_visita.php”.
- *nueva_visita.php*: formulario que permite añadir un comentario.
- *insertar_visita.php*: página que almacena el comentario del visitante. Es una página que no se muestra en el navegador. Una vez introducido un comentario se redirecciona automáticamente a la página “libro_visitas.php”.

Las visitas se han de almacenar en un fichero llamado “visitas.txt”.

Recomendaciones

- Extrae a funciones el código común. Por ejemplo, define una función que lea a un array el contenido del archivo visitas.txt (las funciones fgets() y feof() te podrán ser útiles) y otra que muestre por pantalla la información de dichas visitas.
- Crea una biblioteca con las funciones más utilizadas.

Recuerda también

Seguir las recomendaciones de claridad y separación de código:

- Incluir comentarios, pero no en exceso.
- Poner la mayor parte del código PHP al comienzo de la página web.
- Indentar el código para mayor claridad.

A la hora de programar:

- No intentar hacerlo todo a la vez, separar los problemas. Una vez resuelto uno pasar al siguiente.
- Si salen muchas líneas de código es que algo no está bien. Estudiar si hay una solución más sencilla al mismo problema.
- Cuando se encuentran fallos viene bien poner en puntos clave del código

sentencias echo "
DEPURACIÓN: $variable
" para ver el valor que toman las variables durante la ejecución de la página.
- Estas sentencias se borran o comentan cuando la página esté terminada.

Archivos a entregar

- Código fuente de la web (páginas con la programación).

Práctica 4. PHP y las bases de datos

Para poder realizar los ejercicios de esta práctica se necesitan dos bases de datos que se generan a través de estos dos scripts:

baloncesto.sql

```
CREATE DATABASE IF NOT EXISTS baloncesto; USE baloncesto;
CREATE TABLE IF NOT EXISTS jugadores (
        id INT PRIMARY KEY AUTO_INCREMENT, nombre CHAR(30),
        posicion CHAR(30), partidos INT, puntos float,
        rebotes float, asistencias float);

INSERT INTO jugadores (nombre, posicion, partidos, puntos, rebotes, asistencias)
VALUES ("Valero","base",24,5.2,1.7,9.8),
        ("Juanfran","base",29,6.1,0.8,5.8),
        ("Montilla","escolta",19,11.7,2.7,2.4),
        ("Rodriguez","escolta",23,17.1,1.8,3.7),
        ("Stipes","escolta",31,8.5,3.1,0.9),
        ("Montes","alero",32,13.1,4.6,4.1),
        ("Volkov","ala pivot",11,4.3,5.6,1.3),
        ("Suarez","ala pivot",24,6.9,4.8,4.5),
        ("Carter","ala pivot",26,26.1,9.1,1.8),
        ("Graham","pivot",17,2.1,8.4,0.7),
        ("Cesar","pivot",8,3.1,6.8,0.7),
        ("Jofre","alero",0,0,0,0),
        ("Lehman","ala pivot",0,0,0,0),
        ("Stevenson","pivot",0,0,0,0);
```

fruteria.sql

```
CREATE DATABASE IF NOT EXISTS fruteria;
USE fruteria;
CREATE TABLE IF NOT EXISTS precios (
        id INT PRIMARY KEY AUTO_INCREMENT, fruta CHAR(50),
        precio_kg float,
        temporada CHAR(50));

INSERT INTO precios (fruta, precio_kg, temporada) VALUES
        ("Judias",3.50,"primavera"),
        ("Patatas",0.40,"anual"),
        ("Tomates",1.00,"anual"),
        ("Manzanas",1.20,"invierno"),
        ("Uvas",2.50,"otono"),
        ("melones",0.80,"verano"),
        ("naranjas",1.50,"invierno");
```

Práctica 1

Usando la base de datos fruteria, hacer una página web que muestre una lista de los productos que están a la venta, “listar_fruta.php”.

Práctica 2

Hacer una página web que muestre las estadísticas del equipo de baloncesto. Ha de estar compuesta por las siguientes páginas.

- *equipo.php*: listado de los nombres de los jugadores ordenados por posiciones de juego. Cada nombre tiene un enlace a ficha.php
- *ficha.php*: la ficha de un jugador con sus estadísticas personales.
- *partidos.php*: muestra un listado de los jugadores ordenados por partidos jugados.
- *puntos.php*: muestra un listado de los jugadores ordenados de por puntos anotados por partido.
- *rebotes.php*: muestra un listado de los jugadores ordenados de por rebotes capturados por partido.
- *asistencias.php*: muestra un listado de los jugadores ordenados de por asistencias dadas por partido.

Práctica 3

Estaría bien que se pudieran gestionar las fichas desde la propia web. Para ello hay que agregar los formularios:

nueva_ficha.php: formulario para añadir un jugador y sus datos.

guardar_ficha.php: el script que guarda los datos que provienen del script anterior.

borrar_ficha.php: el script que borra la ficha de un jugador.

Práctica 4

Mejorar la práctica 1 para que se puedan gestionar datos.

- Añadir un formulario que permita añadir frutas, “nueva_fruta.php”. Y una página que lo inserte en la base de datos, “guardar_fruta.php”.
- Programar también un sistema para que se puedan borrar frutas del catálogo, “borrar_fruta.php”.
- Poner un último formulario que permita modificar el precio de una fruta. Primero se han de leer los datos actuales, mostrar en el formulario (editar_fruta.php), modificar y guardarlos en la base de datos mediante un UPDATE (mod_fruta.php).

Práctica 5

Se podría añadir una mejora a la página del equipo de baloncesto que consistiera en un formulario para subir el acta de un partido, y que la aplicación recogiera esa acta, y actualizara las estadísticas de los jugadores.

El acta es un archivo que contiene una línea por cada jugador que ha participado con los siguientes datos (los jugadores que no figuran en el acta es porque no han salido al campo):

nombre del jugador, puntos, rebotes, asistencias;

Un fichero de acta podría ser el siguiente:

Valero, 17, 6, 4;
Juanfran, 8, 1, 11;
Montilla, 12, 4, 2;
Volkov, 7, 7, 0;
Carter, 2, 4, 0;
Stipes, 23, 0, 3;
Cesar, 9, 2, 2;
Montes, 11, 3, 1;

Vamos a detallar los pasos que vamos a seguir.

- Un formulario “subir_acta.php” que permite subir un archivo con un acta.
- El destino de la página “subir_acta.php” es la página “recoger_acta.php”. Ésta realiza todo el trabajo.
  - Primero guarda el archivo temporal en una carpeta actas.
  - Segundo abre el archivo con el acta y lee los datos y los guarda en variables.
  - Por cada jugador que aparece en el acta lee los datos de la base de datos.
  - Calcula la estadística actualizada para cada categoría. Por ejemplo:

```
<?php
// Codigo anterior…
$partidos_ahora = $partidos_antes + 1;
$puntos_antes = $puntos_por_partido_antes * $partidos_antes;
$puntos_ahora = $puntos_antes + $puntos_ultimo_partido;
$puntos_por_partido_ahora = $puntos_ahora / $partidos_ahora;
// Codigo posterior…
?>
```

- Finalmente actualiza los datos de cada jugador en la base de datos.

Es altamente recomendable usar arrays asociativos para almacenar los datos y extraer el código que se usa varias veces a funciones para que el código quede más limpio y ordenado.