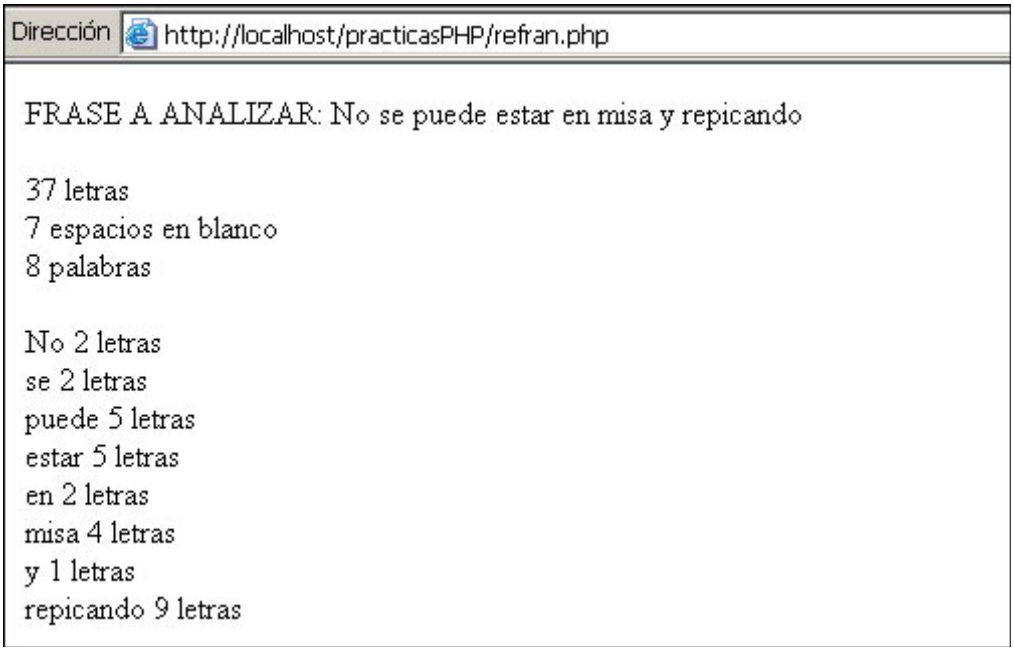
PRÁCTICA 3. CONCEPTOS AVANZADOS

Utilizar el sistema de nombres de la segunda sesión, salvo que el ejercicio indique lo contrario. De igual modo, comprimir todos los ficheros de la entrega en un único archivo zip o rar de nombre *p3_nombrealumno*.

Práctica 3.1. 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 3.2. 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 3.3. 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()*:

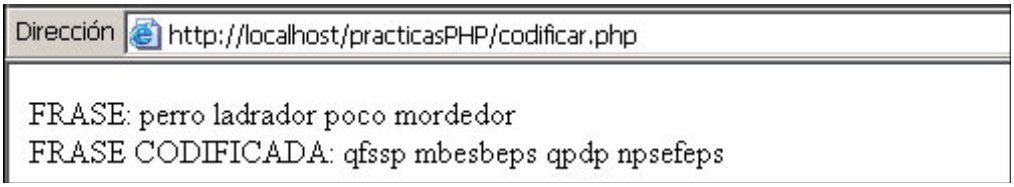
```php
<?php
  echo ord("7") . "<br>"; // Vale 55, luego es un número
  echo ord("c") . "<br>"; // Vale 99, luego es una letra
  echo ord("-") . "<br>"; // Vale 45, ni letra ni número
?>
```

Práctica 3.4. Mejora el ejemplo 3.1.7 para comprobar que la dirección de email no contiene espacios en blanco.

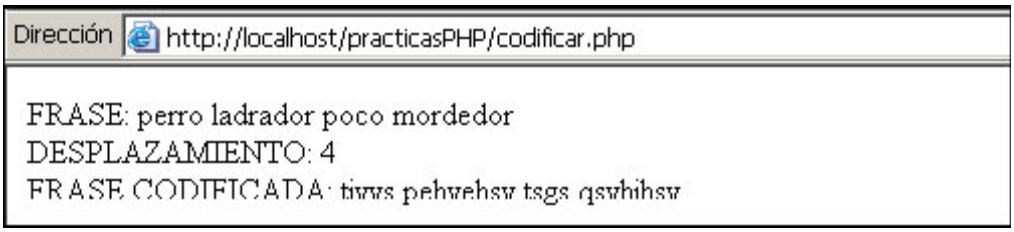
Práctica 3.5. Haz un array indexado numéricamente que contenga las letras del abecedario en minúsculas. Luego recórrelo, y muéstralo por pantalla cada letra del array, primero en minúsculas y luego en mayúsculas.

Práctica 3.6. 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 3.7. 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 3.8. 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 3.9. Vuelve a programar la práctica 2.8 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 2.8.

Práctica 3.10. 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 3.9 para mostrar el tiquet de compras en euros y en pesetas haciendo la conversión con estas funciones.

Práctica 3.11. 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 3.12. Extrae el código del ejercicio 3.7 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 3.13. 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 3.14. 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 3.15. 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 3.12 muestre el resultado de la frase codificada en la página "*codificado.php*".

Práctica 3.16. Amplia el ejercicio 3.4.5 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 3.17. 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 3.18. 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 3.19. 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 3.20. 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.
- Si en algún momento no se sabe por donde seguir en *Internet* hay muchos ejemplos que pueden resultar útiles.
- No dudar en consultar al tutor las dudas que se tengan.

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.