Ejercicio 1)

```
<?php
function doble($i) { return $i*2; }$a = TRUE; $b = "xyz"; $c = 'xyz'; $d = 12; echo
gettype($a); echo gettype($b); echo gettype($c);
echo gettype($d);
if (is_int($d)) {
$d += 4; }
if (is_string($a)) {
          echo "Cadena: $a"; }
$d = $a ? ++$d : $d*3;
$f = doble($d++);
$g = $f += 10;
echo $a, $b, $c, $d, $f , $g;
 ?>
```

1. Variables y su tipo

- $a: Tipo booleano (TRUE)
- $b: Tipo string ("xyz")
- $c: Tipo string ('xyz')
- $d: Tipo entero (12)
- $f: Inicialmente entero, ya que será el resultado de la función doble($d++) que devolverá un valor entero.
- $g: Inicialmente entero, ya que será el resultado de $f += 10, donde $f es un entero.

2. Operadores

- Asignación: =
- Concatenación (implícita en echo): ,
- Operador ternario: ? :
- Operador incremento: ++
- Operadores aritméticos: * (multiplicación), += (suma y asignación)

3. Funciones y sus parámetros

- doble($i): Función que recibe un parámetro $i y devuelve su doble ($i * 2).
- gettype(): Función integrada de PHP que devuelve el tipo de una variable, con un parámetro que es la variable a evaluar.
- is_int(): Función que verifica si una variable es de tipo entero, recibe un parámetro que es la variable a verificar.
- is_string(): Función que verifica si una variable es de tipo string, recibe un parámetro que es la variable a verificar.

- echo: Función que imprime en pantalla, con parámetros que pueden ser valores o variables.

4. Estructuras de control

- if: Se usa en decisiones condicionales:
  -if (is_int($d)) { $d += 4; }: Si $d es entero, entonces se le suma 4.
  -if (is_string($a)) { echo "Cadena: $a"; }: Si $a es un string, imprime "Cadena: $a".
- Operador ternario: $d = $a ? ++$d : $d*3;: Si $a es verdadero, incrementa $d; de lo contrario, multiplica $d por 3.

5. Salida por pantalla

- ➔ gettype($a): $a es booleano, así que imprime "boolean".
- ➔ gettype($b): $b es string, imprime "string".
- ➔ gettype($c): $c es string, imprime "string".
- ➔ gettype($d): $d es entero, imprime "integer".
- ➔ if (is_int($d)): Como $d es entero (12), le suma 4, por lo tanto, ahora $d vale 16.
- ➔ if (is_string($a)): Como $a es booleano y no string, no imprime nada en esta línea.
- ➔ $d = $a ? ++$d : $d*3;: Como $a es TRUE, incrementa $d en 1. Ahora, $d es 17.
- ➔ $f = doble($d++);: Se llama a la función doble(17), que devuelve 34, y luego se incrementa $d, por lo que $d ahora es 18 y $f es 34.
- ➔ $g = $f += 10;: Se suma 10 a $f, por lo tanto $f ahora es 44 y $g también es 44.
- ➔ echo $a, $b, $c, $d, $f, $g;:

Esto imprime:

$a es TRUE, así que imprime 1 (PHP convierte TRUE a 1).
$b es "xyz", lo imprime.
$c es 'xyz', lo imprime.
$d es 18, lo imprime.
$f es 44, lo imprime.
$g es 44, lo imprime.

Salida completa: booleanstringstringinteger1xyzxyz184444

Ejercicio 2)

a) **Código 1**:

```
<?php
$i = 1;
while ($i <= 10) {
```

```
    print $i++;
}
?>
```

Este código imprime números del 1 al 10.

Código 2:

```
<?php
$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
?>
```

Este código es casi idéntico al código 1, pero utiliza la sintaxis alternativa de while (while: y endwhile; en lugar de llaves {}). Da el mismo resultado.

Código 3:

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

Este código también imprime números del 1 al 10 solo que la sentencia do while asegura que el bucle se ejecute al menos una vez, en cambio, los códigos anteriores podrían no ejecutarse si la sentencia $i <= 10 fuera falsa.

b) Código 1:

```
<?php
for ($i = 1; $i <= 10; $i++) {
    print $i;
}
?>
```

Inicialización: $i = 1.
Condición: Mientras $i <= 10, el bucle se ejecuta.
Incremento: Después de cada iteración, se incrementa $i con $i++.

Imprime los números del 1 al 10.

Código 2:

```
<?php
for ($i = 1; $i <= 10; print $i, $i++) ;
?>
```

Este código también imprime los números del 1 al 10, pero $i++ se evalúa después de la impresión.

Código 3:

```
<?php
for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}
?>
```

Este código también imprime los números del 1 al 10, pero además agrega un break para salir del bucle cuando $i>10

Código 4:

```
<?php
$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}
?>
```

El bucle es infinito (for (;;)) pero hay un if que detiene la ejecución cuando $i > 10. Imprime los números del 1 al 10.

c)

Código 1:

```
<?php
…
…
if ($i == 0) {
    print "i equals 0";
} elseif ($i == 1) {
    print "i equals 1";
```

```
} elseif ($i == 2) {
    print "i equals 2";
}
?>
```

Este es un bloque condicional if-elseif:

- Se evalúa primero si $i == 0. Si es verdadero, imprime "i equals 0".
- Si $i != 0, evalúa si $i == 1. Si es verdadero, imprime "i equals 1".
- Si $i != 1, evalúa si $i == 2. Si es verdadero, imprime "i equals 2".
- Si ninguna de las condiciones es verdadera, no hace nada (no hay una condición else final).

Código 2:

```
<?php
…
...
switch ($i) {
    case 0:
        print "i equals 0";
        break;
    case 1:
        print "i equals 1";
        break;
    case 2:
        print "i equals 2";
        break;
}
?>
```

El bloque switch evalúa $i directamente, este código realiza lo mismo que el anterior. Agrega un break para salir de cada case.

Ejercicio 3:

a)

```
<html>
<head><title>Documento 1</title></head>
<body>
<?php
echo "<table width = 90% border = '1' >";
$row = 5;
$col = 2;
```

```
for ($r = 1; $r <= $row; $r++) {
echo "<tr>";
for ($c = 1; $c <= $col;$c++) {
echo "<td> </td>\n";
} echo "</tr>\n";
}
echo "</table>\n";
?>
</body></html>
```

Este es un documento HTML básico con código PHP dentro. Genera una tabla con 5 filas y 2 columnas.

Creación de la tabla:

- echo "<table width = 90% border = '1' >";: Se genera el código HTML para crear una tabla con un ancho del 90% del contenedor y un borde de 1 píxel alrededor de las celdas.

Definición de filas y columnas:

- $row = 5; Se define que la tabla tendrá 5 filas.
- $col = 2; Se define que la tabla tendrá 2 columnas por cada fila.

Generación de las filas:

- El primer bucle for ($r = 1; $r <= $row; $r++) { recorre desde 1 hasta 5 (porque $row = 5), es decir, genera 5 filas.
- Dentro del bucle, echo "<tr>"; genera el código HTML de apertura para cada fila (<tr>).

Generación de las celdas dentro de cada fila:

- El segundo bucle for ($c = 1; $c <= $col; $c++) { recorre desde 1 hasta 2 (porque $col = 2), generando 2 celdas por fila.
- Dentro de este bucle, echo "<td> </td>\n"; genera una celda vacía (<td>) con un espacio en blanco ( ), seguido de una nueva línea (\n para legibilidad).

Cierre de filas y tabla:

- echo "</tr>\n"; cierra cada fila (</tr>) después de que se hayan generado las celdas.
- Después de que el bucle ha terminado, echo "</table>\n"; cierra la tabla HTML.

b)

```
<html>
<head><title>Documento 2</title></head>
<body>
<?php
```

```
if (!isset($_POST['submit'])) {
?>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
Edad: <input name="age" size="2">
<input type="submit" name="submit" value="Ir">
</form>
<?php
}
else {
$age = $_POST['age'];
if ($age >= 21) {
echo 'Mayor de edad';
}
else {
echo 'Menor de edad';
}
}
?>
</body></html>
```

Documento HTML con formulario HTML y código PHP que maneja la lógica de envío y respuesta.

Primera condición: if (!isset($_POST['submit'])) {

- Este bloque comprueba si el formulario ha sido enviado. Si no ha sido enviado (es decir, si no se ha definido $_POST['submit']), entonces el formulario se muestra al usuario.

Formulario HTML:

- <form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">: El formulario envía la información usando el método POST y el atributo action hace que el formulario se envíe al mismo archivo (usando $_SERVER['PHP_SELF'] para obtener el nombre del archivo actual).
- <input name="age" size="2">: Se presenta un campo de texto donde el usuario puede introducir su edad. El tamaño del campo es de 2 caracteres.
- <input type="submit" name="submit" value="Ir">: Este es un botón para enviar el formulario. Cuando se presiona, se envían los datos al servidor.

Cuando el formulario es enviado (es decir, cuando $_POST['submit'] está definido), se ejecuta este bloque de código:

```
else {
   $age = $_POST['age'];
   if ($age >= 21) {
      echo 'Mayor de edad';
```

```
    }
    else {
        echo 'Menor de edad';
    }
}
```

Este bloque realiza lo siguiente:
Recuperación de datos: $age = $_POST['age'];

- La variable $age obtiene el valor introducido por el usuario en el campo de texto age.

Condición:

- Si $age >= 21, se imprime "Mayor de edad".
- Si $age < 21, se imprime "Menor de edad".

Nota: No se valida que ingrese un valor(puede ingresar null).

Ejercicio 4)

Si el archivo datos.php contiene el código que sigue:

```
<?php
 $color = 'blanco';
 $flor = 'clavel';
?>
```

Indicar las salidas que produce el siguiente código. Justificar.

```
<?php
echo "El $flor $color \n";
 include 'datos.php';
 echo " El $flor $color";
?>
```

El archivo datos.php simplemente define las variables $color, $flor
Luego el siguiente código intenta imprimir las variables aun no declaradas, por lo tanto imprimirá “El” (no da error)
Al incluir el archivo datos.php y luego imprimir, la salida será “El clavel blanco”

Ejercicio 5)

Analizar el siguiente ejemplo: Contador de visitas a una página web
contador.php

```
<?
// Archivo para acumular el número de visitas
$archivo = "contador.dat";
// Abrir el archivo para lectura
$abrir = fopen($archivo, "r");
```

Se abre el archivo contador.dat en modo lectura ("r"), permitiendo leer el contenido actual del archivo.

```
// Leer el contenido del archivo
$cont = fread($abrir, filesize($archivo));
```

Se lee el contenido completo del archivo utilizando fread(). El valor leído es el número de visitas actuales, y se almacena en la variable $cont.

```
// Cerrar el archivo
fclose($abrir);
```

El archivo se cierra después de leer su contenido.

```
// Abrir nuevamente el archivo para escritura
$abrir = fopen($archivo, "w");
```

Se abre nuevamente el archivo, pero esta vez en modo escritura ("w"), lo que permite escribir en el archivo y reemplazar el contenido anterior.

```
// Agregar 1 visita
$cont = $cont + 1;
```

Se incrementa el contador en 1.

```
// Guardar la modificación
$guardar = fwrite($abrir, $cont);
```

El nuevo valor del contador se escribe en el archivo `contador.dat`, reemplazando el valor anterior.

```
// Cerrar el archivo
fclose($abrir);
```

El archivo se cierra nuevamente después de escribir el nuevo valor.

```
// Mostrar el total de visitas
echo "<font face='arial' size='3'>Cantidad de visitas:".$cont."</font>";
?>
```

Finalmente, el valor de $cont se muestra en la página web, indicando el número total de visitas.

visitas.php

```
<!-- Página que va a contener al contador de visitas -->
<html>
<head></head>
<body>
<? include("contador.php")?>
</body>
</html>
```

Esta página web incluye el archivo contador.php utilizando la función include(). Cada vez que se carga esta página (visitas.php), el archivo contador.php se ejecuta, incrementando el contador en una unidad y mostrando el número de visitas.

PHP: arrays, funciones

Ejercicio 1)

Indicar si los siguientes códigos son equivalentes.

Código 1:

```
<?php
$a = array(
   'color' => 'rojo',
   'sabor' => 'dulce',
   'forma' => 'redonda',
   'nombre' => 'manzana',
   4
);
?>
```

-Se crea un array asociativo usando la función array().
-Los elementos 'color', 'sabor', 'forma' y 'nombre' son pares clave-valor.
-El valor 4 es agregado al final del array sin una clave específica, por lo que PHP lo asignará a la clave numérica 0 (ya que es el primer índice libre en un array vacío).

Código 2:

```
<?php
$a['color'] = 'rojo';
$a['sabor'] = 'dulce';
$a['forma'] = 'redonda';
$a['nombre'] = 'manzana';
$a[] = 4;
?>
```

Este código es equivalente al anterior solo que se crean los elementos del array uno por uno asignando valores a las claves.

Ejercicio 2)

a)

```
<?php
$matriz = array("x" => "bar", 12 => true);
```

//Se define el array $matriz, la clave x tiene el valor “bar” y la clave 12 el valor “true”

```
echo $matriz["x"];
```

//Se imprime el valor asociado a la clave x, que es “bar”

```
echo $matriz[12];
```
//Se imprime el valor asociado a la clave 12, que es “true” pero al imprimir un booleano PHP lo imprime como 1
```
?>
```

Salida final: bar1

b)

```
<?php
$matriz = array("unamatriz" => array(6 => 5, 13 => 9, "a" => 42));
```
// elemento con la clave "unamatriz" contiene otro array asociativo con tres elementos:
-Clave 6 con valor 5
-Clave 13 con valor 9
-Clave "a" con valor 42
```
echo $matriz["unamatriz"][6];
```
//Se imprime el valor asociado a la clave 6, que es 5.
```
echo $matriz["unamatriz"][13];
```
//Se imprime el valor asociado a la clave 13, que es 9.
```
echo $matriz["unamatriz"]["a"];
```
//Se imprime el valor asociado a la clave a, que es 42.
```
?>
```

Salida final: 5942

c)

```
<?php
$matriz = array(5 => 1, 12 => 2);
```
//Define el array que contiene:
-Clave 5 con valor 1
-Clave 12 con valor 2
```
$matriz[] = 56;
```
//Agrega el valor 56 al array sin especificar clave, PHP asigna automáticamente la primer clave disponible(en este caso 0)
```
$matriz["x"] = 42; unset($matriz[5]); unset($matriz);
```
//Agrega el valor 42 con clave “x”, luego borra el elemento con clave 5(con unset) y luego elimina el array completo con unset($matriz)
```
?>
```

Este código no produce ninguna salida.

Ejercicio 3)

a)

```
<?php
$fun = getdate();
echo "Has entrado en esta pagina a las $fun[hours] horas, con $fun[minutes] minutos y $fun[seconds] segundos, del $fun[mday]/$fun[mon]/$fun[year]";
?>
```

Este código da un mensaje que contiene la hora, minutos, segundos, día, mes y año en el que se accedió a la página. Toma la fecha actual cuando se ejecuta con la función getdate().

b)

```
<?php
function sumar($sumando1, $sumando2) {
   $suma = $sumando1 + $sumando2;
   echo $sumando1 . "+" . $sumando2 . "=" . $suma;
}
sumar(5, 6);
?>
```

Este código define la función que recibe 2 parámetros($sumando1,$sumando2).
Al llamar a la función sumar(5,6) la salida será 5+6=11

Ejercicio 4)

```
function comprobar_nombre_usuario($nombre_usuario){
//compruebo que el tamaño del string sea válido.
if (strlen($nombre_usuario)<3 || strlen($nombre_usuario)>20){
```

//Verifica si el tamaño del nombre es menor que 3 o mayor que 20 caracteres. Si esto ocurre, el nombre se considera inválido.

```
 echo $nombre_usuario . " no es válido<br>";
 return false;
}
//compruebo que los caracteres sean los permitidos $permitidos =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789-_";
for ($i=0; $i<strlen($nombre_usuario); $i++){
if (strpos($permitidos, substr($nombre_usuario,$i,1))===false){
echo $nombre_usuario . " no es válido<br>";
return false; }
```

//Define una lista de caracteres permitidos y recorre cada carácter del nombre de usuario. Si algún carácter no está en la lista, el nombre se considera inválido.

```
 }
```

```
echo $nombre_usuario . " es válido<br>";
 return true;
//El nombre se considera válido si pasa ambas comprobaciones.
}
```