

ENTORNOS GRÁFICOS

TRABAJO PRÁCTICO PHP

**Comisión**: 3 (4E03)

**Integrantes**:

|  |  |
| --- | --- |
| Legajo | Apellido y Nombre |
| 45882 | Romaniuk, Federico Nicolás |
| 45878 | Galdeano, Gonzalo |
| 46472 | Raselli, Gianfranco |

# PRÁCTICA N°4

**PHP: variables, tipos, operadores, expresiones, estructuras de control PHP: arrays, funciones**

# 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;

?>

En el siguiente código identificar:

* las variables y su tipo
* los operadores
* las funciones y sus parámetros
* las estructuras de control
* cuál es la salida por pantalla

# Resolución:

1. **Variables y su tipo:**
   1. $a = Boolean

$b = String

$c = String

$d = Integer

$g=Integer

$f=Integer

$i = Integer

# Operadores:

* 1. \* -->binario

ii. += --> binario

iii. $a ? ++$d : $d\*3 -->ternario

iv. ++ -->unario

# Funciones:

* 1. doble():
     1. Parametro : $i
  2. gettype():
     1. parametros ($a,$b,$c,$d)
  3. is\_int():
     1. Parametros: $d
  4. is\_string():
     1. Parametros $a

# Estructuras de control:

* 1. if
  2. asignaciones
  3. llamada a función

# Salida por pantalla

**Ejercicio 2:**

Indicar si los siguientes códigos son equivalentes. a)

<?php

for ($i = 1; $i <= 10; print $i, $i++) ;

?>

<?php

$i = 1;

while ($i <= 10) { print $i++;

}

?>

<?php

$i = 1;

while ($i <= 10): print $i;

$i++; endwhile;

?>

b)

<?php

$i = 0; do {

print ++$i;

} while ($i<10);

?>

<?php

for ($i = 1; $i <= 10; $i++) { print $i;

}

?>

<?php

for ($i = 1; ;$i++) { if ($i > 10) {

<?php

$i = 1; for (;;) {

if ($i > 10) { break;

}

print $i;

$i++;

}

?>

break;

}

print $i;

}

?>

c)

<?php

…

…

if ($i == 0) {

print "i equals 0";

} elseif ($i == 1) { print "i equals 1";

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

}

?>

<?php

…

…

switch ($i) { case 0:

print "i equals 0"; break;

case 1:

print "i equals 1"; break;

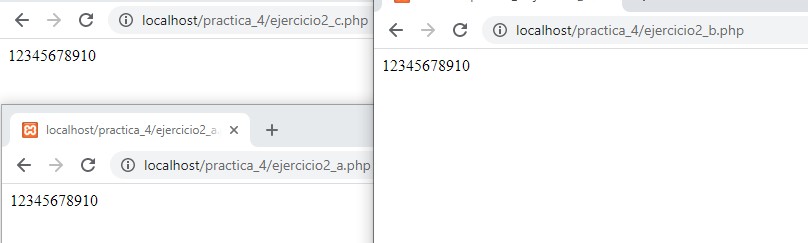
case 2:

print "i equals 2"; break;

}

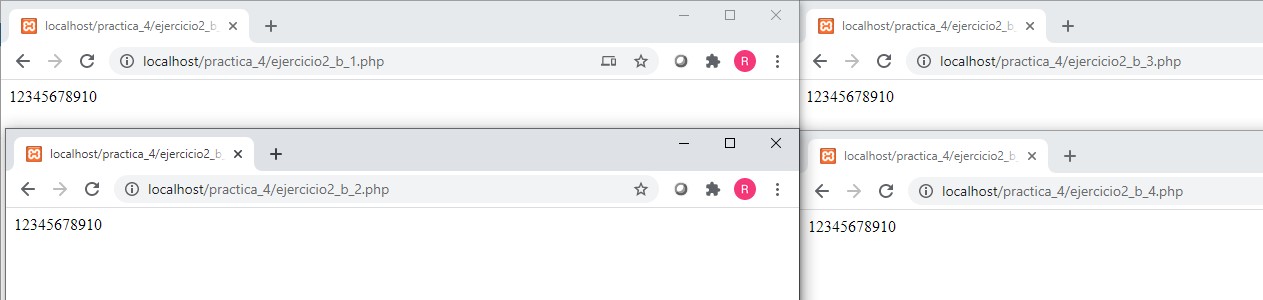
?>

# Resolución a:



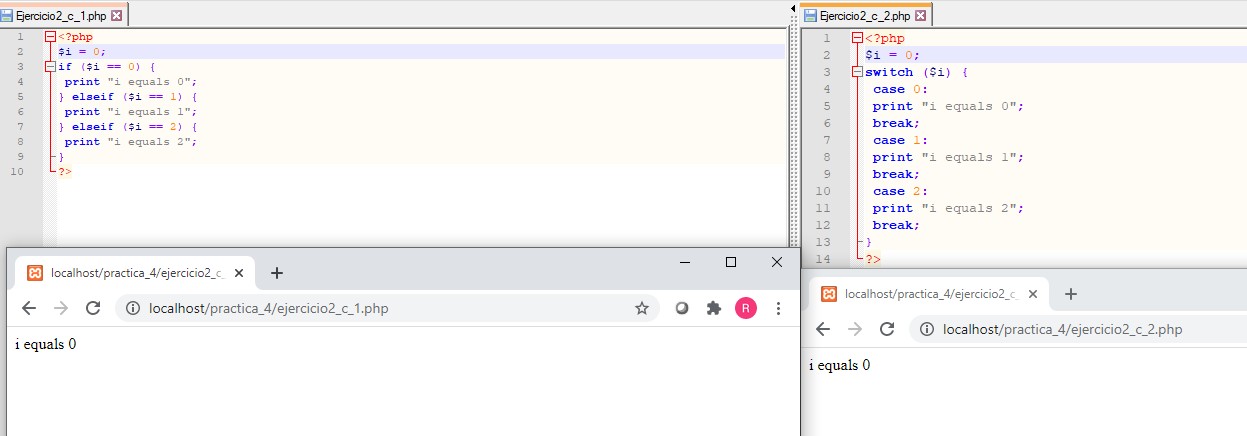
Visualmente podemos ver que el resultado en los 3 casos es el mismo. Si bien la lógica aplicada es diferente, y algunos tienen mejor performance que otro, el resultado es lo mismo.

# Resolución b:



Este caso al igual que el anterior, el resultado en pantalla es en los 4 códigos el mismo. Obviamente, la forma de trabajar internamente es diferente. Algunos presentan mejor performance que otros.

# Resolución C:



En ambos casos instanciamos la variable $i en 0. Nuevamente el resultado por pantalla es el mismo. Aunque el código y el funcionamiento, pueden trabajar de forma diferente.

# Ejercicio 3:

Explicar para qué se utiliza el siguiente código: 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>&nbsp;</td>\n";

} echo "</tr>\n";

}

echo "</table>\n";

?>

</body></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';

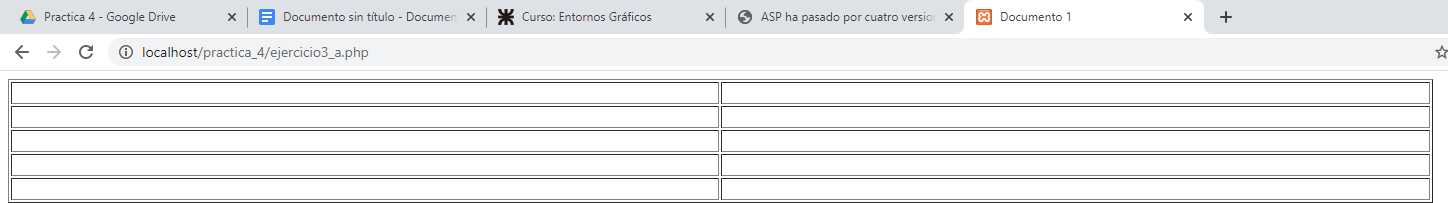
}

}

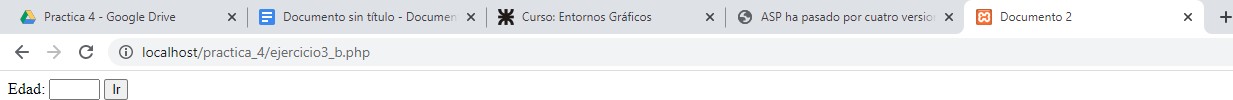
?>

</body></html>

# Resolución a:

* Salida por pantalla:
* En este caso el codigo nos permite:
  + Crear un archivo Php donde el nombre del head sea Documento 1
  + Crear una tabla con sentencia for utilizando php. Donde la fila se especifica que sean 5 y las columnas 2.

# Resolución b:

* Salida por pantalla:
  + En este caso se utiliza la variable $\_Post que permite recoger los datos de un formulario después que se completa.
  + Permite utilizar un formulario input y un boton para desencadenar otra acción después de llenar el formulario.
  + Muestra “mayor de edad” si edad >=21, menor de edad si <=21.

# 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";

?>

# Resolución:

* En primer lugar, el programa nos informa de que no reconoce las variables flor y color. Ya que no están definidas e instanciadas en ninguna otra parte del programa.
* Después de utilizar la sentencia include, que nos permite llamar a un archivo donde se encuentran estas variables, podemos utilizarlas como si estuvieran dentro del programa.
* Esto sucede porque php necesita una instanciación de las variables utilizadas en el código, antes de que sean llamadas.

# Ejercicio 5:

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

# contador.php

<?

// Archivo para acumular el numero de visitas

$archivo = "contador.dat";

// Abrir el archivo para lectura

$abrir = fopen($archivo, "r");

// Leer el contenido del archivo

$cont = fread($abrir, filesize($archivo));

// Cerrar el archivo fclose($abrir);

// Abrir nuevamente el archivo para escritura

$abrir = fopen($archivo, "w");

// Agregar 1 visita

$cont = $cont + 1;

// Guardar la modificación

$guardar = fwrite($abrir, $cont);

// Cerrar el archivo fclose($abrir);

// Mostrar el total de visitas

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

?>

**visitas.php**

<!-- Página que va a contener al contador de visitas -->

<html>

<head></head>

<body>

<? include("contador.php")?>

</body>

</html>

En la misma carpeta, crear el archivo de texto **contador.dat**, con el valor inicial del contador y con permisos de lectura y escritura.

# Resolución:

A contador.php y vistas.php le hace falta php luego de <? para funcionar.

Vistas.php invoca al archivo.php y cuando corre por primera vez crea un archivo contador.dat que cuenta la cantidad de visitas que tiene el mismo. Luego de almacenarlo, lo cierra y vuelve a abrirlo nuevamente, pero con permisos de escritura, donde aumenta en uno el valor y escribe el archivo y muestra la cantidad de visitas aumentada.

# PHP: arrays, funciones

**Ejercicio 1:**

Indicar si los siguientes códigos son equivalentes.

<?php

$a = array( 'color' => 'rojo', 'sabor' => 'dulce', 'forma' => 'redonda', 'nombre' => 'manzana', 4

);

?>

<?php

$a['color'] = 'rojo';

$a['sabor'] = 'dulce';

$a['forma'] = 'redonda';

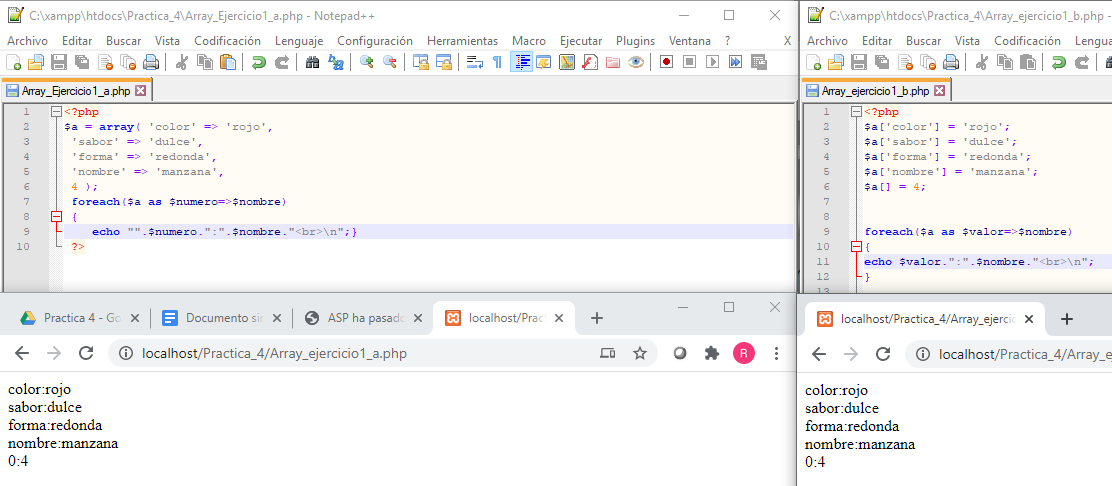
$a['nombre'] = 'manzana';

$a[] = 4;

?>

# Resolución:

Salida por pantalla:



Como se puede observar en las imágenes, en ambos casos el resultado en pantalla es exactamente el mismo cuando recorremos el arreglo. Por lo cual, considero que son equivalentes.

# Ejercicio 2:

En cada caso, indicar las salidas correspondientes: a)

<?php

$matriz = array("x" => "bar", 12 => true); echo $matriz["x"];

echo $matriz[12];

?>

b)

<?php

$matriz = array("unamatriz" => array(6 => 5, 13 => 9, "a" => 42));

echo $matriz["unamatriz"][6]; echo

$matriz["unamatriz"][13];

c)

<?php

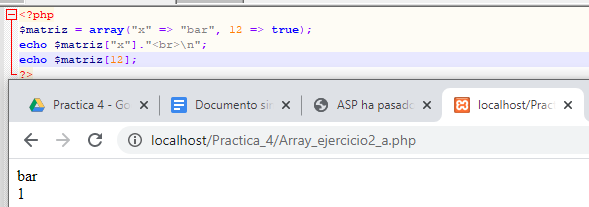
$matriz = array(5 => 1, 12 => 2);

$matriz[] = 56;

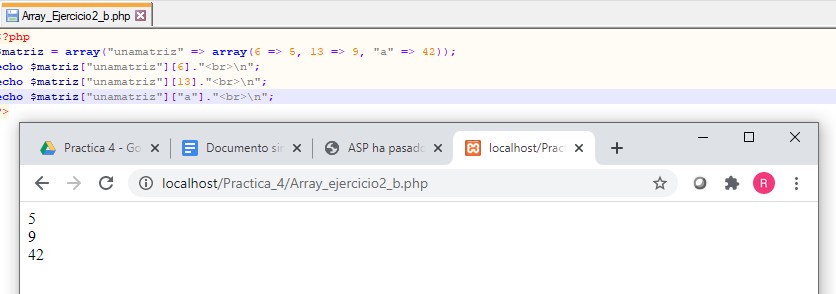
$matriz["x"] = 42; unset($matriz[5]); unset($matriz);

?>

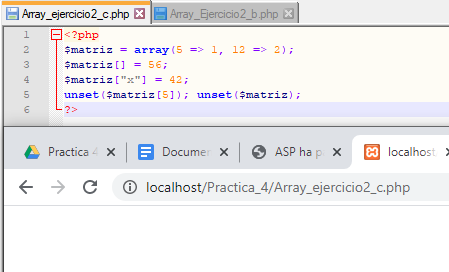
# Resolución 2.a)



**Resolución 2.b)**



# Resolución 2.c)



**Ejercicio 3:**

En cada caso, indicar las salidas correspondientes a)

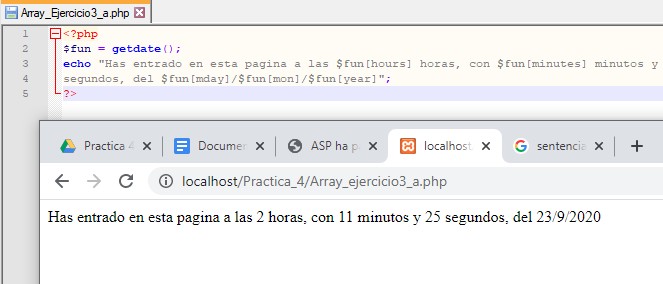
<?php

$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]";

?>

# Resolución 3.a)



La función getdate() nos devuelve los valores de hora y fecha, en este caso del servidor que estamos usando en Xampp.

b)

<?php

function sumar($sumando1,$sumando2){

$suma=$sumando1+$sumando2;

echo $sumando1."+".$sumando2."=".$suma;

}

sumar(5,6);

?>

# Resolución 3.b)

En este caso la función sumar recibe los parámetros 5 y 6 en su llamada.

# Ejercicio 4:

Analizar la siguiente función, y escribir un script para probar su funcionamiento:

function comprobar\_nombre\_usuario($nombre\_usuario){

//compruebo que el tamaño del string sea válido.

if (strlen($nombre\_usuario)<3 || strlen($nombre\_usuario)>20){ 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;

}

}

echo $nombre\_usuario . " es válido<br>"; return true;

}

# Resolución:

La función cuenta con dos sentencias de control, la primera un if que determina si el parámetro enviado a la función, en este caso, cualquier nombre de usuario enviado, cumpla con una longitud determinada, mayor a 3 y menor a 20 caracteres. En el caso de no cumplir muestra el mensaje de que el nombre de usuario ingresado no es válido. La función retorna un false.

Si el nombre de usuario pasa la primera sentencia se ejecuta la segunda sentencia de control que es un for anidado a un if, el mismo verifica que el nombre de usuario enviado cumpla con los caracteres alfanuméricos. De no cumplir lo notifica. La función retorna un false.

De cumplir con el tamaño y los caracteres permitido se muestra un mensaje indicando que el nombre de usuario es válido. La función retorna un true.

