Entorno Gráficos

Práctica 4

PHP - Variables y funciones

**Profesores:**

* Prof. Ing. Daniela Díaz
* Prof. Ing. Julián Butti

**Integrantes:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Legajo** | **E-Mail** |
| Castillo, Gonzalo | 43098 | gonzalo07.xc@gmail.com |
| Pinacca, Franco | 43133 | fpinacca@gmail.com |
| Diez, Juan Pablo | 44158 | jpdok98@gmail.com |
| Mateucci, Nicolas | 42868 | nico.mateucci@gmail.com |
| Ulla, David | 42111 | davoulla@gmail.com |

[Primera parte: variables, tipos, operadores, expresiones, estructuras de control](#_lpxkzlsg8qzc)

[Ejercicio 1:](#_9ot8tr8csvip)

[Ejercicio 2: Indicar si los siguientes códigos son equivalentes.](#_og2rdzy67l2d)

[Ejercicio 3](#_th99gfna8k6i)

[Ejercicio 4](#_dx0f71my8zwe)

[Ejercicio 5](#_ygvdb53bc03i)

[Segunda Parte: Manejo de Arreglos](#_gz3i3m5ap2kj)

[Ejercicio 1](#_og31umdqcw4x)

[Ejercicio 2](#_l0o3qzi2res3)

[Ejercicio 3](#_bkb8z6m7df)

[Ejercicio 4](#_8in0sdrmgmrz)

## Primera parte: variables, tipos, operadores, expresiones, estructuras de control

### Ejercicio 1:

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.

<?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;

?>

Resolución:

|  |  |  |
| --- | --- | --- |
| **Variable** | **Tipo** | **Ejemplo** |
| **$a = TRUE true = True** | **Booleana.**  TRUE = true = 1 | **echo $a**  **Salida: 1** |
| **$b = “xyz”** | **String**  Reemplaza variables que puede haber dentro del string | **echo “La variable a vale $a”**  **Salida: La variable a vale 1** |
| **$c = ‘xyz’** | **String literal**  No reemplaza variables | **echo “La variable a vale $a”**  **Salida: La variable a vale $a** |
| **$d = 12** | **Integer** |  |
| **$f** | **Integer** |  |
| **$g** | **Integer** |  |

Operadores:

|  |  |  |
| --- | --- | --- |
| **Operador** | **Descripciòn** | **Tipo** |
| **=** | Operador de asignación | Binario |
| **\*** | Operador de multiplicación | Binario |
| **+=** | Equivale a sumar la misma variable más un argumento dado. | Binario |
| **,** | Concatenar cadenas para imprimir por pantalla. | Binario |
| **++$arg** | Incrementa el valor en 1 y luego realiza la asignación | Unario |
| **$arg++** | Primero realiza la asignación y luego incrementa el valor en 1. | Unario |

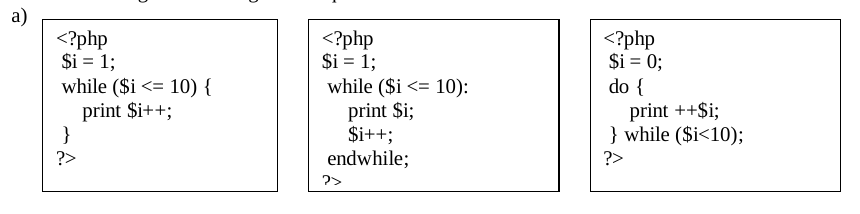
Funciones:

|  |  |
| --- | --- |
| **Función** | **Tipo** |
| **echo** | Imprime por pantalla |
| **print** | Imprime por pantalla |
| **doble($arg)** | Devuelve el doble del valor pasado como argumento. |
| **is\_int($arg)** | Devuelve 0 o 1 si el argumento es entero. |
| **is\_string($arg)** | Devuelve 0 o 1 si el argumento es una cadena. |

Estructuras de control:

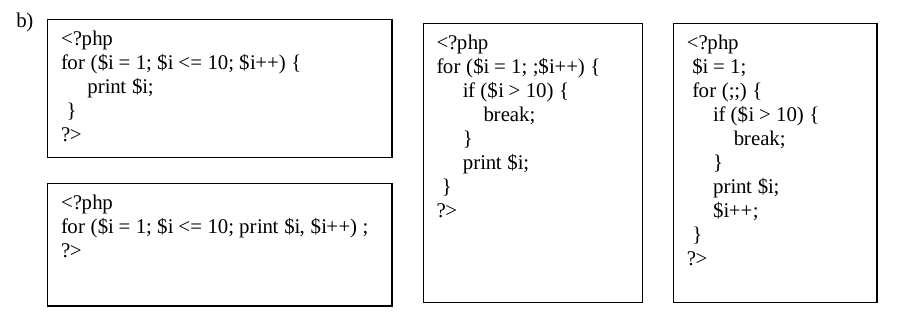
|  |  |
| --- | --- |
| **Estructura** | **Tipo** |
| **if{}** | Sentencia if tradicional |
| **$a1? a2: a3** | Sentencia if abreviada. Operador ternario. |

### Ejercicio 2: Indicar si los siguientes códigos son equivalentes.

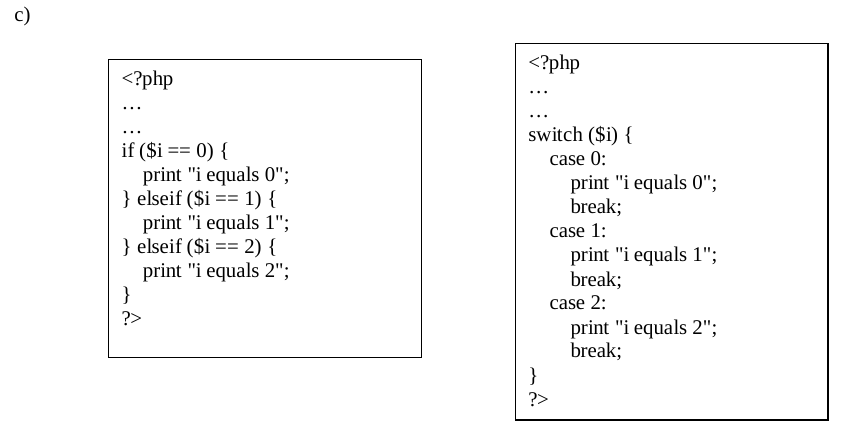


**Son equivalentes**. En los 3 casos se imprimen los números del 1 al 10 por pantalla.

En el primer caso se imprime la variable $i dentro de la estructura while y luego se la incrementa en 1. En el segundo caso sucede lo mismo, solo que se separa la linea de impresión por pantalla y la de incremento, además de que se usa otra sintaxis de la estructura while. En el tercer caso, se usa la estructura do-while y aunque la variable $i se inicializa en 0 y no en uno, el operador de incremento usado primero incremente la variable y luego se la imprime con el la función print, por lo que también se genera la misma salida que los casos anteriores.



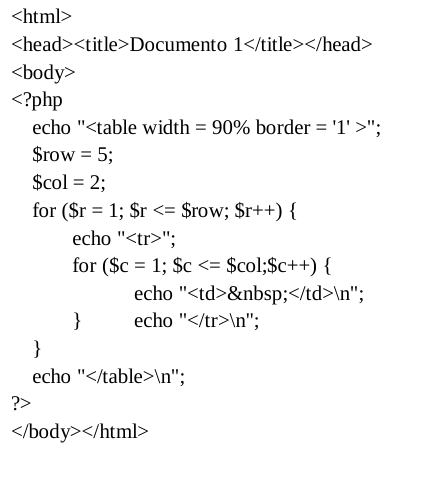
**Son equivalentes**. En los 4 casos se imprimen los números del 1 al 10 por pantalla. En el caso 1 se utiliza una estructura for tradicional. En el segundo se agrega la impresión por pantalla en el tercer argumento del for. En el caso 3 se utiliza la condición if dentro del for para cortar el repeticion cuando la variable $i llega a 10 y en el cuarto caso se excluyen los tres argumentos del for, inicializando afuera de esta la variable $i y también utilizando una condición if para cortarlo.



**Son equivalentes**. La única diferencia es en las estructuras de control utilizadas.

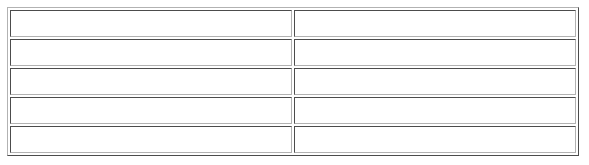
### Ejercicio 3

a)



En un documento html se crea (utilizando PHP) una tabla que ocupa el 90% del espacio y tiene un borde de 1. Luego se definen la cantidad de filas y columnas en dos variables (5 y 2 respectivamente). Por último, utilizando dos estructuras for anidadas, se crean 5 filas en la tabla y a cada celda se le inserta el carácter **&nbsp; (No Blocking Space).**

La salida por pantalla sería la siguiente:

****

### 

b)



Consideraciones:

* **isset**: Determina si una variable está definida y no es **null**. [Link a doc](https://www.php.net/manual/es/function.isset.php).
* **$\_POST**: Arreglo de variables pasadas al script actual a través del método POST. [Link a doc.](https://www.php.net/manual/es/reserved.variables.post)

La condición:

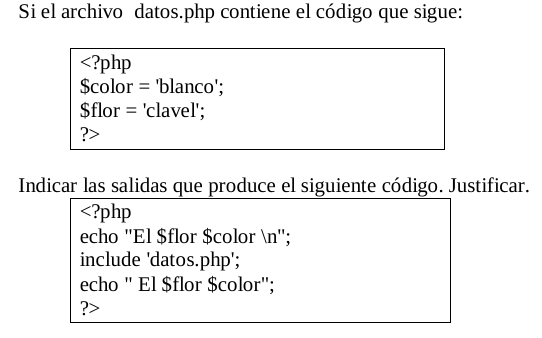
isset($\_POST['submit'])

Evalúa si en el documento actual se hizo el submit del formulario (Apretando el botón “submit”).

Entonces, al correr el script, como todavía no se hizo el submit, esa condición está en falso, por lo que al negarlo (con el operador **“!”**), el if ingresa por el camino verdadero,

Luego, cuando el usuario ingresa la edad y apreta el botón “Ir”, se envía el submit, que es mismo script se encarga de manejar con la condición false del if.

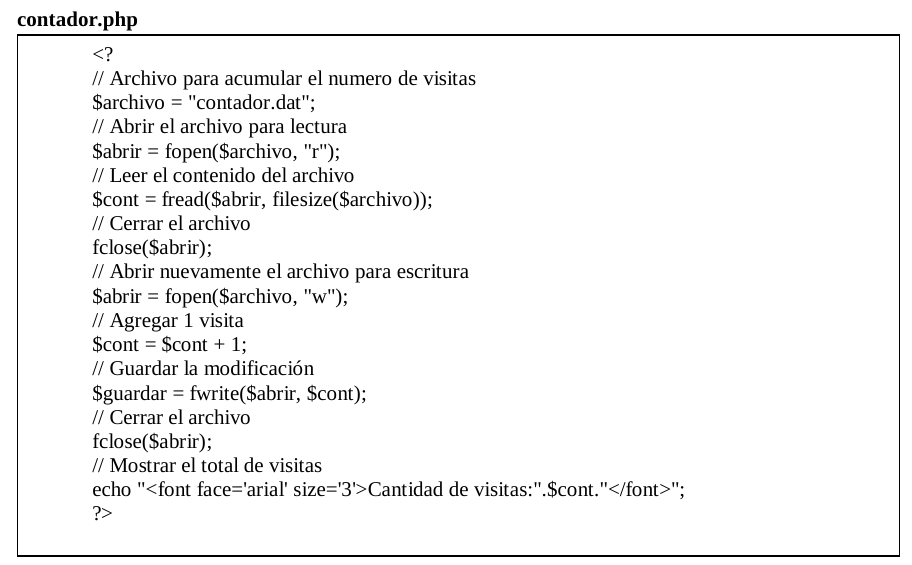
### Ejercicio 4

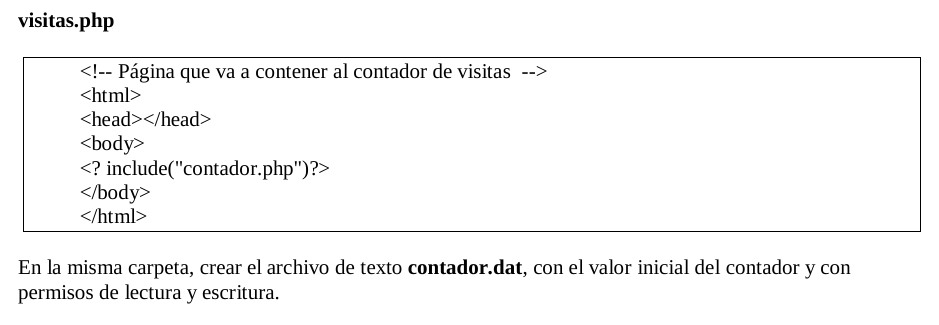


Inicialmente las variables no están inicializadas, por lo que la salida del primer **echo es “El”**. Luego, con la línea **“include ‘datos.php’** se inicializan las variables $color y $flor, por lo que la salida del **echo es “El clavel blanco”**.

### Ejercicio 5

Analizar el siguiente programa:





El programa cuenta la cantidad de visitas a partir del número guardado en el disco dentro del archivo **contador.dat**. Luego, cada vez que una persona solicita la página, el programa lee este archivo en modo lectura para obtener el número de visitas actual y luego lo abre en modo escritura, para incrementar el contador en 1 y luego volver a guardarlo en disco.

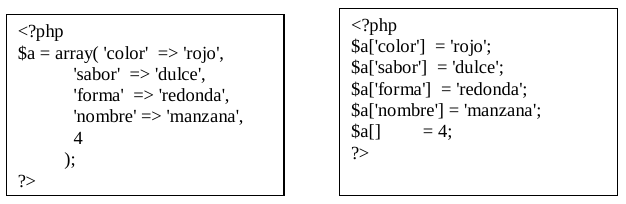
Finalmente, se muestra por pantalla con la instrucción echo la cantidad de visitas hasta el momento.

Es importante destacar, que en caso de que el navegador no tenga permisos de modificación de los archivos que están en el servidor, este no podrá ejecutar la función **fwrite** para guardar el nuevo número de visitas. En cambio, sí podrá ejecutar la función fopen, que no necesita permisos especiales.

## Segunda Parte: Manejo de Arreglos

### Ejercicio 1

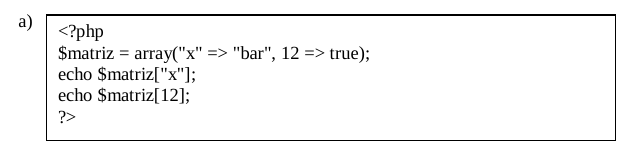
Indicar si los siguientes códigos son equivalentes



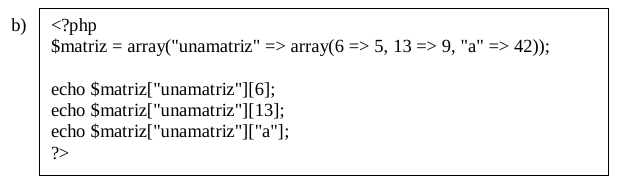
Si, ambos son equivalentes y son dos formas distintas de crear un arreglo en PHP.

### Ejercicio 2

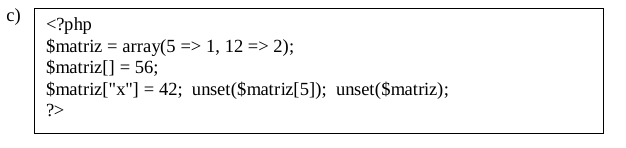
Indicar las salidas



**La salida es: “bar” 1.**

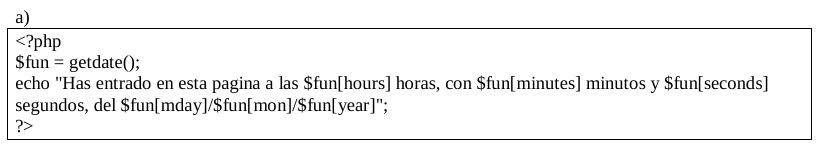


**La salida es: 5 9 42**



Este código no genera salidas, solo crea el arreglo y luego lo borra con la función **unset()**.

### Ejercicio 3

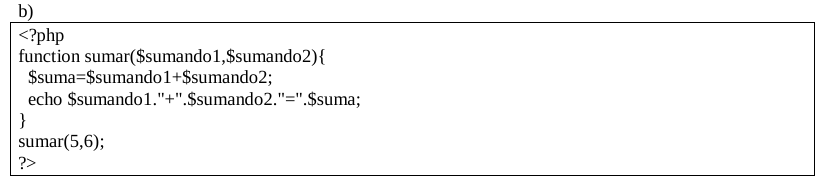


La función **getdate()** devuelve un arreglo con la siguiente estructura:

Array ( [seconds] => 2 [minutes] => 4 [hours] => 22 [mday] => 13 [wday] => 4 [mon] => 5 [year] => 2021 [yday] => 132 [weekday] => Thursday [month] => May [0] => 1620936242 )

Luego, la salida del programa corresponde con la información del día y la hora en que llama a la función.

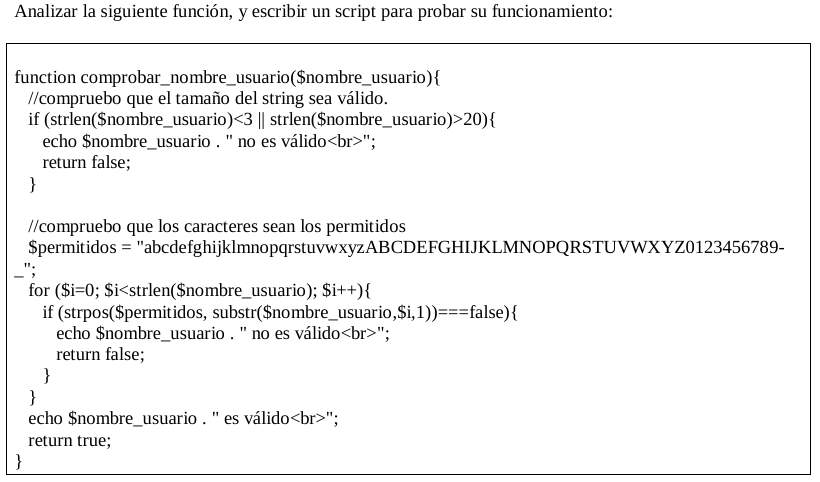
En este caso: “Has entrado en esta pagina a las 22 horas, con 4 minutos y 2 segundos, del 13/5/2021 ”



El programa crea una función que recibe dos parámetros, realiza la suma de ellos guardandolos en una variable y luego concatena las 3 variables con el operador . (punto) para mostrarlas por pantalla.

La salida es: “5+6=11”

### Ejercicio 4



$nombres = array(1 => "nicomate22", 2 => 'davoulla@gmail', 3 => 'an', 4 => 'nicolasmateucci@hotmail.com');

for($i = 1; $i <= sizeof($nombres); $i++){

echo "<br>Prueba numero $i:";

comprobar\_nombre\_usuario($nombres[$i]);

}

La salida del programa es:

Prueba numero 1:nicomate22 es válido

Prueba numero 2:davoulla@gmail no es válido

Prueba numero 3:an no es válido

Prueba numero 4:nicolasmateucci@hotmail.com no es válido