PHP: variables, tipos, operadores, expresiones, estructuras de control

1)

- *a* es de tipo *boolean.*
- *b* es de tipo *string.*
- *c* es de tipo *string.*
- *d* es de tipo *integer.*
- *f* es de tipo *integer.*
- *g* es de tipo *integer.*
- Operador de asignación =, al momento de crear las variables.
- Función *gettype()*, con las variables como parámetros.
- La estructura de control *if*, cuya condición es la función que verifica si la variable *d* es entero. Da positivo y le asigna a *d* el valor de *d* + 4. En este caso ahora *d* pasa a valer 16.
- La estructura de control *if*, cuya condición es la función que verifica si la variable *a* es un *string*. Da negativo.
- Operador ternario. Se le asigna a la variable *d*, si *a* es *true*, el valor de *d* incrementado en 1. Si es falso, el valor de *d* multiplicado por 3. Como *a* es *true*, *d* pasa a valer ahora 17.
- Operador de asignación =. A *f* le asigna el valor del doble de *d*. Luego, *d* se incrementa en 1. *f* vale 34 y *d* vale 18.
- Operador de asignación =. A *g* se le asigna el valor de *f* sumado en 10. *g* vale 44 y *f* también vale 44.
- Luego muestra el valor de todas las variables.

Salida por pantalla:

```
boolean
string
string
integer
1
xyz
xyz
18
44
44
```

2)

a) Los 3 códigos generan la misma salida por pantalla:

```
12345678910
12345678910
12345678910
```

En los dos primeros, se inicializa la variable *i* en 1, y entra en un *while*, mientras sea menor a 10, muestra y luego incrementa en 1. El tercero es una estructura *do..while*. Se inicializa en cero, y antes de mostrar, se incrementa en 1. Termina cuando *i* es menor a 10 (el 10 se

muestra porque incrementa en 1 y luego muestra. Cuando muestra el 10, entra en el *do..while* porque *i* vale 9).

b) Los 4 códigos generan la misma salida por pantalla:

```
12345678910
12345678910
12345678910
12345678910
```

El primero es una estructura de control *for*, donde se inicializa la variable *i* en 1, corta cuando *i* es menor igual a 10, y va incrementando en 1 a *i* cuando termina de mostrar el valor de *i*.

El segundo es una estructura de control *for*, donde solamente se inicializa en 1 y se incrementa en 1 luego de cada ejecución. No existe una condición de corte dentro del *for*. En cambio, existe una sentencia *if*, que cuando *i* es mayor a 10, se ejecuta un *break*, que automáticamente corta el *for*. En cada iteración va mostrando el valor de *i*.

El tercero es una estructura *for*, pero sin ninguna condición interna. Se inicializa la variable *i* en 1 antes de entrar en el *for*. Muestra *i* y luego incrementa en 1 la variable. Como en el anterior, existe una sentencia *if* que ejecuta un *break* que automáticamente termina el bucle *for*, con condición de que *i* sea mayor a 10.

El cuarto es una estructura *for*, donde se inicializa la variable *i* en 1 y condición de fin del bucle es que *i* sea mayor igual a 10. Al final de cada ejecución, muestra *i* e incrementa en 1 dicha variable.

c) Los 2 códigos generan la misma salida por pantalla, inicializando la variable i en cero (*$i = 0;*):

```
i equals 0
i equals 0
```

En el primero, existe una estructura de control *if...else if...else if* , que según *i* satisfaga la condición de cada *if*, entra y ejecuta el código que tiene. Una vez que encuentra uno, termina la ejecución de toda la estructura *if..else if*.

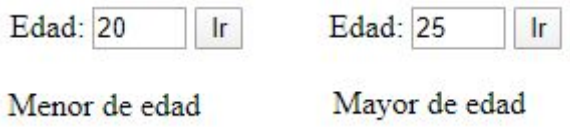
En el segundo, existe una estructura de control *switch*. Según el valor que tenga la variable *i*, entra en algún *case*. Muestra mediante el *print* y corta mediante el *break*.

3)

a) El código genera una tabla en html. Por pantalla se ve lo siguiente:

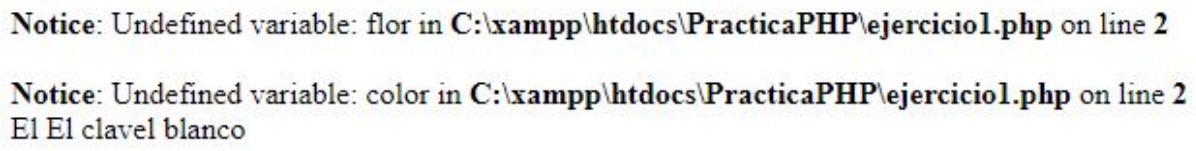
Inicializa las variables *col* y *row*, y genera las filas y tablas mediante *for*. El primer *for* es para las filas y el segundo *for* es para las columnas. Se crea con valor vacío cada bloque (* *).

b) El código genera un campo, donde se ingresa la edad y luego presionando un botón, nos dice si es mayor o menor de edad, según la condición de que la edad ingresada sea mayor o no a 21 años. Crea una variable age, que se valoriza según el valor que ingresemos de edad, todo esto dentro de un formulario. Luego con un *if..else*, imprime en pantalla mayor de edad o menor de edad.



4)

El código muestra lo siguiente en pantalla:



Los dos *Notice* aparecen porque se quiere mostrar 2 variables que, hasta el momento, no están definidas. Luego, ejecuta la sentencia *include 'datos.php';* , y en este momento, por el código que tiene el archivo *datos.php*, la variable *flor* y *color* ya tienen tipo y valor y pueden ser mostradas. Aparecen dos *El*, porque se concatena con lo que sí puede mostrar del primer *echo*.

5)

Incluye un código que se utiliza para saber cuantas veces se visitó una página. La cantidad de veces se almacena en *contador.dat,* que, al momento de ingresar a la página, se abre, lee, se suma en 1 la variable cantidad de visitas y luego se graba en el archivo contador. Existe una salida por pantalla, que nos muestra la cantidad de visitas históricas que ha tenido la página.

PHP: arrays, funciones

1)

Ambos códigos generan un array unidimensional. Ambos códigos generan una parte del array “asociativa”, con partes clave-valor (por ejemplo clave=sabor valor=dulce) y una indexada (por ejemplo $a[0]=4).

2)

a) La salida es “bar1”, ya que lo que se quiere mostrar por pantalla es el valor de los campos *x* y *12* del array *$matriz*. Cuando mostramos el valor del campo llamado *12*, nos muestra 1 porque es el valor de true.

b) La salida es “5942”, ya que lo que se quiere mostrar por pantalla es el valor de los campos *6, 13* y *a*, dentro del array multidimensional *$matriz*. Se accede primero al array dentro del array mayor al que se quiere acceder, en este caso es "*unamatriz*", y luego a los campos dentro de este array.

c) En la salida no se muestra nada, por no tener las sentencias *echo*. Si se las añadimos, nos tira error por querer mostrar un valor nulo. Esto es por la sentencia *unset($matriz);* ya que la función unset elimina todos los valores que tenga la variable *$matriz.* Devuelta, si eliminamos esa sentencia, y solamente dejamos las otras, nos muestra los datos que no sean nulos, los que no hayan sido eliminados por algún *unset*.

3)

a) Se ve por pantalla la hora a la que entramos a la página y la fecha, sumadas 5 horas (desconozco de qué GMT es la hora). La función getdate(), genera un array con campos clave-valor, que contienen datos sobre la fecha (hora, minutos, segundos, dia, mes, año).

b) Nos muestra 5+6=11, ya que ejecuta la función *sumar($sumando1,$sumando2)*, y le pasa como parámetros los valores 5 y 6. Luego calcula la suma de ambos y los muestra en el formato $sumando1."+".$sumando2."=".$suma.

4)

Se ingresa un nombre y se valida de que su longitud esté entre 3 y 20 caracteres, y también que sus caracteres están dentro de los permitidos. Si cumple con ambas condiciones, se considera válido el nombre; caso contrario, inválido.

Para probar su funcionamiento, se codificó lo siguiente:

```
<html>
     <head>
        </head>
<body>

<?php if (!isset($_POST['submit'])) { ?>

<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
 Nombre: <input name="nombre" size="6">
  <input type="submit" name="submit" value="Comprobar">
</form>
<?php
} else {
$nombre_usuario = $_POST['nombre'];
```

```
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;   }

        }
        comprobar_nombre_usuario($nombre_usuario);
?>
</body>
</html>
```

Nos permite ingresar el nombre, y luego de que presionamos el botón *Comprobar*, nos indica si el nombre es válido o no, según posea las características previamente dichas.