Arrays

Programando en PHP

Un pequeño inciso

¿Qué es una función?

- Una función es un **trozo de código** que se almacena bajo un **nombre**.
- Cuando se llama a una función, el código almacenado se ejecuta.
- De una función nos interesa lo que hace, pero NO cómo lo hace.
- Algunas funciones necesitan una serie de parámetros de entrada para funcionar.

¿Qué es una función?

- Hay dos tipos de funciones:
  - Devuelven datos
  - No devuelven datos
- Cuando una función devuelve un dato, dicho dato hay que recogerlo en una variable.
- Una función se puede llamar tantas veces como sea necesario.

¿Qué es una función?

- Las llamadas a una función se hacen de la siguiente forma:

```
Variable_recogida = nombre_funcion (parametros)
```

```
Nombre_funcion (parametros)
```

- En PHP existen muchísimas funciones predefinidas que nos ayudan a realizar las tareas.

- **Php.net**

Introducción

Introducción

INTRODUCCIÓN

- Los arrays son una parte muy importante de cualquier lenguaje de programación.
- Permiten:
  - Manejar grupos de valores relacionados
  - Almacenar múltiples valores en una sola estructura y bajo un mismo nombre.
- Muchas de las funciones de PHP devuelven un array de valores.
- En PHP los arrays están muy ligados a las bases de datos.
- Tipos de arrays:
  - Posicionales
  - Asociativos.

Los arrays posicionales

Arrays posicionales

- Formados por un conjunto de valores ordenados respecto a un índice.
- El índice entero, indica la posición del elemento en el conjunto.
- Formas de asignar un array:
  - La más sencilla → Asignar a cada posición el valor.
  - Utilizando la función array()

ASIGNACIÓN DE ARRAYS POSICIONALES

```php
$array1[0]=12;
$array1[1]="verde";
$array1[2]=25.4;
$array1[3]="vivo";
$array1[]="Riviera";

$array2 = array (12, "verde", 25.4, "vivo", "Riviera");
```

Arrays

Posición	0	1	2	3	4
Array 1	12	verde	25.4	vivo	Riviera
Array 2	12	verde	25.4	vivo	Riviera

Los arrays asociativos

Arrays asociativos

- Formados por un conjunto de valores ordenados respecto a un índice que no es entero si no string.

- Formas de asignar un array:
  - La más sencilla → Asignar a cada posición el valor.
  - Utilizando la función array(). En este caso será necesario indicar el nombre de la posición.

ASIGNACIÓN DE ARRAYS ASOCIATIVOS

```php
<?php
    $array1["posi1"]=12;
    $array1["posi2"]="verde";
    $array1["posi3"]=25.4;
    $array1["posi4"]="vivo";
    $array1["posi5"]="Riviera";

    $array2 = array ("posi1"=>12, "posi2"=>"verde", "posi3"=>25.4,
                     "posi4"=>"vivo", "posi5"=>"Riviera");
```

Arrays

Posición	0	1	2	3	4
Array 1	12	verde	25.4	vivo	Riviera
Array 2	12	verde	25.4	vivo	Riviera

Arrays multidimensionales

Arrays multidimensionales

- PHP nos permite definir arrays multidimensionales mediante la combinación de arrays unidimensionales (tanto posicionales como asociativos)

- Veamos ejemplos:

Asignación de arrays multidimensionales

```php
<?php
    $matriz1[0][0]="Peseta";
    $matriz1[0][1]=166.386;
    $matriz1[1][0]="Dólar";
    $matriz1[1][1]=0.96;

    $matriz2[0] = array ("Peseta", 166.386);
    $matriz2[1] = array ("Dólar", 0.96);

    $matriz3 = array (array ("Peseta", 166.3
```

matrices

	Moneda	Cambio €
$matriz1[0]	Peseta	166.386
$matriz1[1]	Dólar	0.96
$matriz2[0]	Peseta	166.386
$matriz2[1]	Dólar	0.96
$matriz3[0]	Peseta	166.386
$matriz3[1]	Dólar	0.96

Recorrer arrays posicionales

Recorrer arrays posicionales

- Lo más habitual cuando se trabaja con arrays es recorrerlos para obtener sus elementos.
- La forma más sencilla de hacerlo es utilizando bucles.
- Problema: Debemos conocer a priori el tamaño.
- **count (array)**
  - Devuelve el número de elementos que hay en el array.

```php
for ($i=0; $i<count($array); $i++)
{
    echo "$array[$i]";

}
```

```php
<?php
    $a[0] = 1;
    $a[1] = 3;
    $a[2] = 5;
    $result = count($a);
    // $result == 3

    $b[0]  = 7;
    $b[5]  = 9;
    $b[10] = 11;
    $result = count($b);
    // $result == 3

    $result = count(null);
    // $result == 0

    $result = count(false);
    // $result == 1
?>
```

Recorrer arrays asociativos

Recorrer arrays asociativos

- Además de saber el número de elementos que tiene el array, **deberíamos** saber las claves para poder acceder a ellos.

- PHP incluye una serie de funciones que nos van a hacer la vida más fácil a la hora de utilizar arrays asociativos.

Recorrer arrays asociativos

- **array_keys(array)**
  - Devuelve un nuevo array **posicional** con las claves que forman el array.

- **array_values(array)**
  - Devuelve un nuevo array **posicional** con los valores que forman parte del array.

Ejemplo Recorrer

```php
$claves = array_keys($vector1);
$valores = array_values($vector1);
for($i=0; $i<count($claves); $i++)
{
    echo "<tr align='center'><td>".$claves[$i]."</td>";
    echo "<td>".$valores[$i]."</td></tr>";
}
```

Ejercicio

- Crear un array asociativo y utilizando las funciones **array_keys** y **array_values** junto con los bucles necesarios, mostrarlo de la siguiente forma:

Nombre	Altura	Edad	Pelo	Ciudad
Juan	3.5	25	Moreno	Granada

```php
<?php
    $persona = array ('nombre'=>'Juan', 'altura'=>3.5,
                      'edad'=>25, 'pelo'=>'moreno',
                      'ciudad'=>'granada');
    $posiciones = array_keys ($persona);
    $i=0;
    echo "<table border><tr>";
    do
    {
        echo "<td>";
        echo $posiciones[$i];
        echo "</td>";
        $i++;
    }while ($i<count($posiciones));
    echo "</tr>";

    $valores = array_values($persona);
    echo "<tr>";
    $i=0;
    do
    {
        echo "<td>";
        echo $valores[$i];
        echo "</td>";
        $i++;
    }while ($i<count($valores));
    echo "</tr></table>";
?>
```

Recorrer arrays asociativos

- También podemos recorrer un array asociativo utilizando el bucle foreach.
- En este caso la sintaxis cambia un poco:

```
Foreach ($array as $posicion=>$valor)
{
    Sentencias
 }
```

Ejercicio

- Crear un array asociativo y utilizando el bucle foreach, mostrarlo de la siguiente forma:

Nombre	Juan
Altura	3.5
Edad	25
Pelo	Moreno
Ciudad	Granada

```php
<?php

    $a['nombre']='Juan';
    $a['altura']=3.5;
    $a['edad']=25;
    $a['pelo']='Moreno';
    $a['ciudad']='Granada';

    echo "<table border>";

    foreach ($a as $posi=>$valor)
    {
        echo "<tr>";
        echo "<td> $posi </td> <td> $valor </td>";
        echo "</tr>";
    }
    echo "</table>";
?>
```

Ordenar Arrays

Ordenar Arrays

- sort(array)
  - Ordena alfabéticamente los valores.
  - De menor a mayor.
  - Se modifica el array original
  - Se pierde la relación entre índice y valor

- rsort(array)
  - Ordena de forma inversa a sort.

```php
<?php
    $frutas = array("limón", "naranja",
            "platano", "albaricoque");
    sort($frutas);
    foreach ($frutas as $clave => $valor) {
        echo "<td>frutas[" . $clave . "]</td>";
        echo "<td> $valor </td>";
        echo "<tr></tr>";
    }
?>
```

Posicion	Valor
frutas[0]	albaricoque
frutas[1]	limón
frutas[2]	naranja
frutas[3]	platano

Ordenar Arrays

- asort(array)
  - Ordena igual que sort, pero mantiene la relación entre índice y valor.

- arsort(array)
  - Ordena de forma inversa a asort.

Ejemplo Ordenación

Vector sin ordenar

Posición	Valor
0	Madrid
1	Zaragoza
2	Bilbao
3	Valencia
4	Lérida
5	Alicante

Vector ordenado con sort

Posición	Valor
0	Alicante
1	Bilbao
2	Lérida
3	Madrid
4	Valencia
5	Zaragoza

Vector ordenado con asort

clave	Valor
5	Alicante
2	Bilbao
4	Lérida
0	Madrid
3	Valencia
1	Zaragoza

Ordenar Arrays

- ksort(array)
  - Ordena alfanuméricamente las claves de un array de menor a mayor.
  - Mantiene las relaciones entre índice y valor.

- krsort(array)
  - Ordena de forma inversa a ksort.

Ejemplo Ordenación

```
$vector = array ('d'=> 'Madrid', 'c'=> 'Zaragoza',
                 'e'=> 'Bilbao', 'b'=> 'Valencia',
                 'f'=> 'Lérida', 'a'=> 'Alicante');
```

Vector sin ordenar

Posición	Valor
d	Madrid
c	Zaragoza
e	Bilbao
b	Valencia
f	Lérida
a	Alicante

Vector ordenado con ksort

Posición	Valor
a	Alicante
b	Valencia
c	Zaragoza
d	Madrid
e	Bilbao
f	Lérida

Otras funciones de arrays

Otras funciones de arrays

- array_reverse (array)
  - Devuelve el array pasado como parámetro, pero con sus componentes en orden inverso.

Otras funciones de arrays

- array_count_values(array)
  - Devuelve un array en el que:
    - Los índices son los contenidos del array original
    - Los valores asociados son la frecuencia de repetición de dichos valores en el array original.

```php
$vector = array (1, 2, 4, 2, 5, 1, 2, 4, 2, 5, 6);
$vector_count = array_count_values ($vector);
```

Vector resultado de count

claves	Frecuencia
1	2
2	4
4	2
5	2
6	1

Otras funciones de arrays

- in_array(elemento_buscar, array)
  - Nos dice si un elemento está dentro de un array o no.
- compact()
  - Recibe como argumento una lista de variables que han sido definidas previamente.
  - Devuelve un array en el que los índices son los nombres de las variables y el contenido, sus correspondientes valores.

Un pequeño inciso

Números aleatorios

- rand(inf, sup) o mt_rand(inf, sup)
  - Ambas devuelven un número aleatorio entre el límite inferior y el límite superior.
  - En ambos casos hay que inicializar la semilla a partir de la que se van a generar los números:
    - srand(semilla) o mt_srand(semilla)
    - Esto no es necesario a partir de PHP 4.2.0 que ya lo hace de forma automática.

```php
srand(8);
$num_aleatorio = rand(1, 15);
$num_aleatorio2= rand(1, 15);
$num_aleatorio3= rand(10, 80);
```

```php
srand(time());
$num_aleatorio = rand(1, 15);
$num_aleatorio2= rand(1, 15);
$num_aleatorio3= rand(10, 80);
```

Arrays

Programando en PHP