

PHP

Programación: funciones y arrays

Parte II: Programación

- Funciones
- Arrays

FUNCIONES

FUNCIONES

Una función es un subprograma o rutina que realiza una tarea específica y puede devolver un resultado para ser utilizado fuera de ella.

Las funciones se cargan en memoria y SOLO se ejecutan en el momento en que son llamadas

FUNCIONES I

Función que no devuelve ningún valor sin parámetros de entrada:

```
<?php
    function saludar() {
        echo "Hola mundo";
    }
    saludar();
?>
```

FUNCIONES II

Función que no devuelve ningún valor con un parámetro de entrada:

```
<?php
    function saludar($nombre) {
        echo "Hola ".$nombre;
    }
    saludar("Daniel");
?>
```

FUNCIONES III

Función que no devuelve ningún valor con varios parámetros de entrada. Los parámetros irán separados por comas:

```
<?php
    function sumar($numero1, $numero2) {
        echo $numero1+$numero2;
    }
    sumar(6,7);
?>
```

VARIABLES COMO PARÁMETROS DE FUNCIONES

Podemos utilizar variables para informar los parámetros de la función cuando la llamamos:

```
<?php
    function sumar($numero1, $numero2) {
        echo $numero1+$numero2;
    }
    $a=6; $b=7;
    sumar($a, $b);
?>
```

A la función **sumar()** le enviamos los valores de las variables **\$a** y **\$b**

EJERCICIO I

Crea una función que recibe el ancho y el alto de un rectángulo y calcula su perímetro ($\text{ancho} \times 2 + \text{alto} \times 2$).

El ancho y el alto lo capturaremos en un formulario

Ejemplo: `calculaPerimetro($alto, $ancho);`

| | | | | | |
|---------------------|--------------------------------|--|------|--------------------------------|--|
| ancho | <input type="text" value="4"/> | | alto | <input type="text" value="3"/> | <input type="button" value="Enviar consulta"/> |
| El perimetro es: 14 | | | | | |

FUNCIONES IV

Función que devuelve un valor para ser utilizado fuera de ella:

La función `saludar()` devuelve el valor 'buenos dias' fuera de la función pero no la variable `texto`

```
<?php
function calcularPerimetro($an, $al) {
    $resultado=$an*2+$al*2;
    //devolver el resultado fuera la función
    return $resultado;
}

$res=calcularPerimetro($ancho, $alto);
echo $res;

?>
```

EJERCICIO

Reescribir el ejercicio de la calculadora pero ahora definiremos funciones para las operaciones aritméticas.

Después de realizar el cálculo, no se borrarán las cifras ni la operación que se han utilizado para el mismo

El resultado se mostrará en un campo al lado del botón de enviar

The image shows a calculator interface with two input fields, an operator, and a result field. The first input field contains the number '4', followed by a plus sign operator, and the second input field contains the number '2'. To the right of these is a button labeled 'Enviar consulta'. To the right of the button is a result field containing the number '2'.

NOTA 1: Para 'recordar' las cifras tendremos que asignar al campo **value** la variable php que utilizamos al recuperar los valores del formulario

NOTA 2 Para 'recordar' la opción de la combo habrá que asignar el atributo **selected** a la opción que hayamos escogido

VARIABLES LOCALES Y GLOBALES I

En php, por defecto, las variables que se definen FUERA de las funciones son variables GLOBALES y sólo se ven en un ámbito global

Las variables definidas DENTRO de las funciones son variables LOCALES y sólo se ven en el ámbito local donde han sido creadas (dentro de la propia función)

VARIABLES LOCALES Y GLOBALES II

Veamos unos ejemplos:

```
<?php
    $a = 1;
    verVariables();
    function verVariables() {
        echo $a; //no se visualizará nada ya que php busca la variable local $a
    }
    echo $a; //se visualizará 1 ya que php busca la variable global $a
?>
```

```
<?php
    $a = 1;
    verVariables();
    function verVariables() {
        $a = 2;
        echo $a; //se visualizará 2 ya que php busca la variable local $a
    }
    echo $a; //se visualizará 1 ya que php busca la variable global $a
?>
```

VARIABLES LOCALES Y GLOBALES III

```
<?php
    verVariables();
    function verVariables() {
        $a = 3;
        echo $a; //se visualizará 3 ya que php busca la variable local $a
    }
    echo $a; //no se visualizará nada ya que php busca la variable global $a
?>
```

Si queremos, desde dentro de una función, tener acceso a las variables globales tenemos que incluir dentro de ésta la palabra clave **global**:

```
global $a, $b;
```

VARIABLES LOCALES Y GLOBALES IV

Ejemplo de uso de **global**:

```
<?php
    $a=1;
    verVariables();
    function verVariables() {
        global $a; //hacemos que la variable global $a pueda usarse dentro de la función
        echo $a; //se visualizará 1 ya que php busca la variable global $a
        $a = 3; //modificamos la variable global $a y le asignamos un nuevo valor
    }
    echo $a; //¿qué valor se visualizará fuera de la función?
?>
```

EJERCICIO COSTE HOTEL I

Vamos a confeccionar un formulario para recoger la información que necesitaremos para calcular el precio de un viaje.

1.- Creamos un formulario con el siguiente aspecto (aproximadamente)

La combo tendrá los siguientes Valores:

- Madrid
- París
- Los Angeles
- Roma

The image shows a web form with the following elements:

- Número de noches:** A text input field with a vertical scrollbar on the right.
- Destino:** A dropdown menu with a blue border. The current selection is "Madrid", which is highlighted in blue. The dropdown list is open, showing the following options: "Madrid", "Madrid", "Paris", "Los Angeles", and "Roma".
- Días alquiler coche:** A text input field with a vertical scrollbar on the right.
- Enviar consulta:** A button with a grey background and black text.
- Coste total:** A text input field containing the number "0".

EJERCICIO COSTE HOTEL II

Dentro del documento anterior vamos a crear una función php llamada **costeHotel** con estas características:

- Recibirá como parámetro un número de noches que previamente habremos recuperado del formulario
- Calculará el precio total a pagar (consideraremos un precio fijo de 60€ por noche).
- Retornará este precio fuera de la función
- Recogemos el precio fuera de la función en una variable llamada **costeHotel** y la enviamos al campo '**Coste Total**' del formulario

Ejemplo para 5 noches:

| | |
|--|-------------------------------------|
| Número de noches: | <input type="text" value="5"/> |
| Destino: | <input type="text" value="Madrid"/> |
| Días alquiler coche: | <input type="text"/> |
| <input type="button" value="Enviar consulta"/> | |
| Coste total: | <input type="text" value="300"/> |

EJERCICIO COSTE HOTEL III

Continuando el ejercicio anterior, después de la función ya escrita añadiremos otra función llamada **costeAvion** con estas características:.

- Recibirá como parámetro de entrada el nombre de la ciudad que seleccionaremos de la combo
- Devolverá fuera de la función un coste en función de dicho parámetro.
- Recogemos el precio fuera de la función en una variable llamada **costeAvion**, la sumamos a la variable **costeHotel** y enviamos el resultado al campo '**Coste Total**' del formulario

Los costes por ciudad son los siguientes: "Madrid": 150; "París": 250; "Los Angeles": 450 ".; "Roma": 200

EJERCICIO COSTE HOTEL IV

Continuando el ejercicio anterior, después de las dos funciones ya escritas añadiremos otra función llamada **costeCoche**:

- Recibirá como parámetro de entrada un número de días de alquiler
- Devolverá fuera de la función un coste en función de dicho parámetro pero teniendo en cuenta los siguientes aspectos:
 - Cada día de alquiler cuesta 40 €
 - Si se alquila por tres o más días se obtiene un descuento de 20 €.
 - Si se alquila por 7 días o más se obtiene un descuento de 50 € (no acumulable con los 20 € de haber alquilado por más de 3 días)
- Recogemos el precio fuera de la función en una variable llamada **costeCoche** la sumamos a las variables **costeHotel** y **costeAvion** y enviamos el resultado al campo '**Coste Total**' del formulario

EJERCICIO COSTE HOTEL V

Por último añadiremos el código necesario para no perder los valores de los campos ni la combo del formulario una vez realizado el cálculo del coste del viaje

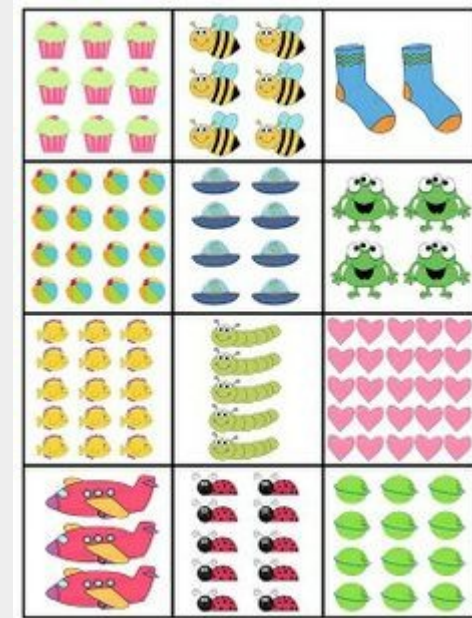
| | |
|--|------------------------------------|
| Número de noches: | <input type="text" value="4"/> |
| Destino: | <input type="text" value="Paris"/> |
| Días alquiler coche: | <input type="text" value="3"/> |
| <input type="button" value="Enviar consulta"/> | |
| Coste total: | <input type="text" value="910"/> |

ARRAYS

ARRAYS I

Un array es una variable que contiene un conjunto de valores:

```
<?php
    $cosas[0] = 9;
    $cosas[1] = 6;
    $cosas[2] = 2;
    $cosas[3] = 16;
    $cosas[4] = 8;
    ...
    echo "contenido de 1: $cosas[0]";
<?
```



ARRAYS II

Otras formas de declarar un array:

```
$cosas = array(9, 6, 2, 6, 8, ...)  
$cosas = [9, 6, 2, 6, 8, ...]
```

Con texto:

```
$dias = array('lunes', 'martes', 'miercoles', ...)  
$dias = ['lunes', 'martes', 'miercoles', ...]
```

Mixtos:

```
$a=3; $b='a';  
$mixto = array(3, 'texto1', 5, $a, $b, true, ...)  
$mixto = [3, 'texto1', 5, $a, $b, true, ...]
```

PRINT_R Y VAR_DUMP

Con la instrucción `print_r()` podemos mostrar el contenido de un array:

```
<?php
$array = array(9, 'texto1', 2, 6, 8);
print_r($array);
?>
```

```
Array ( [0] => 9 [1] => texto1 [2] => 2 [3] => 6 [4] => 8 )
```

Con la instrucción `var_dump()` mostramos el contenido y los detalles de un array:

```
<?php
$array = array(9, 'texto1', 2, 6, 8);
var_dump($array);
?>
```

```
array(5) { [0]=> int(9) [1]=> string(6) "texto1" [2]=> int(2) [3]=> int(6) [4]=> int(8) }
```


AÑADIR ELEMENTOS

Si tenemos el siguiente array:

```
$array = array("rojo", "azul", "verde");
```

Para añadir elementos al final del array:

```
array_push($array, 'amarillo', 'blanco');
```

Para añadir elementos al principio del array:

```
array_unshift($array, 'negro', 'rosa');
```

```
Array ( [0] => rojo [1] => azul [2] => verde )
```

```
Array ( [0] => rojo [1] => azul [2] => verde [3] => amarillo [4] => blanco )
```

```
Array ( [0] => negro [1] => rosa [2] => rojo [3] => azul [4] => verde [5] => amarillo [6] => blanco )
```

BORRAR ELEMENTOS

Si tenemos el siguiente array:

```
$array = array("negro", "rojo", "azul", "verde", "amarillo", "blanco");
```

Para borrar el elemento del final del array:

```
array_pop($array); (opcionalmente $ultimo=array_pop($array))
```

Para borrar el elemento del principio:

```
array_shift($array); (opcionalmente $primero=array_shift($array))
```

```
Array ( [0] => negro [1] => rojo [2] => azul [3] => verde [4] => amarillo [5] => blanco )  
Array ( [0] => negro [1] => rojo [2] => azul [3] => verde [4] => amarillo )  
Array ( [0] => rojo [1] => azul [2] => verde [3] => amarillo )
```

ELIMINAR CUALQUIER ELEMENTO

Si tenemos el siguiente array:

```
$array = array("negro", "rojo", "azul", "verde", "amarillo", "blanco");
```

Si queremos eliminar el tercer elemento (azul):

```
unset($array[2]);
```

```
Array ( [0] => negro [1] => rojo [2] => azul [3] => verde [4] => amarillo [5] => blanco )  
Array ( [0] => negro [1] => rojo [3] => verde [4] => amarillo [5] => blanco )
```

NOTA: Podemos ver como el elemento `$array[2]` ha desaparecido, es decir, NO SE REORGANIZAN las claves del array

EJERCICIO

- Guardar cinco nombres de persona en un array php
- Eliminar el segundo elemento sin que se reorganicen los índices
- Añadir un nuevo nombre al principio del array
- Añadir dos nuevos nombres al final

NOTA: Después de cada paso se mostrará el array en pantalla

ARRAY_SPLICE

Con `array_splice()` podemos eliminar varios elementos del array, en cualquier posición y reordenando los índices:

```
$array = array("negro", "rojo", "azul", "verde", "amarillo", "blanco");
```

Si queremos eliminar del segundo al cuarto elemento (rojo al verde):

```
array_splice($array, 1, 3); //array_splice($matriz, desde, tamaño)
```

```
Array ( [0] => negro [1] => rojo [2] => azul [3] => verde [4] => amarillo [5] => blanco )  
Array ( [0] => negro [1] => amarillo [2] => blanco )
```

NOTA 1: Podemos ver como se han reorganizado las claves

NOTA 2: Con `$nuevoArray=array_splice($array, 1, 3)` guardamos los elementos eliminados en un array nuevo

CONVERTIR UNA LISTA EN ARRAY

Para convertir una lista de elementos en un array

```
$lista = "negro, rojo, azul, verde, amarillo, blanco";  
$array = explode(",", $lista);
```

Para convertir un array en una lista (opción 1):

```
$array = array("negro", "rojo", "azul", "verde", "amarillo", "blanco");  
$lista = implode(",", $array);
```

EJERCICIO

Crear una lista con los 12 meses del año, mostrarla en pantalla con `echo` y, posteriormente, convertirla en array

Mostrar el array con `print_r` en pantalla

ARRAYS ASOCIATIVOS I

Supongamos el siguiente array:

```
$datosPersona[0] = 'Luis Rodriguez';  
$datosPersona[1] = '44444444P';  
$datosPersona[2] = 'c/ Pepe Gotera s/n';  
$datosPersona[3] = 'Barcelona';  
$datosPersona[4] = '08000';
```

```
echo 'nombre y apellidos: '.$datosPersona[0];
```


ARRAYS ASOCIATIVOS II

php, permite la creación de arrays asociativos, en donde el índice del mismo lo podemos substituir por un texto que identifique el contenido de esa posición del array:

Ejemplo:

```
$datosPersona=array();
```

```
$datosPersona['nombre'] = 'Luis Rodriguez';
```

```
$datosPersona['nif'] = '44444444P';
```

```
$datosPersona['direccion'] = 'c/ Pepe Gotera s/n';
```

```
$datosPersona['poblacion'] = 'Barcelona';
```

```
$datosPersona['cp'] = '08000';
```

```
echo 'nombre y apellidos: '.$datosPersona['nombre'];
```

Podemos acceder a cualquier elemento del array por el nombre del índice.

ARRAYS ASOCIATIVOS III

Para crear el mismo array asociativo utilizando el método abreviado:

```
$datosPersona =  
array('nombre' => 'Luis Rodriguez', 'nif' => '44444444P', 'direccion' => 'c/  
Pepe Gotera s/n', 'poblacion' => 'Barcelona', 'cp' => '08000');  
  
echo 'nombre y apellidos: '.$datosPersona['nombre'];
```

NOTA: Si usamos `implode()` o `explode()` con arrays asociativos perderemos las claves

EJERCICIO

Crear un array asociativo con la relación País-Capital de cinco estados del mundo:

Ej:

Francia – París

Ecuador – Quito

Gran Bretaña – Londres

Alemania – Berlin

Argentina – Buenos Aires

Mostrar el array por pantalla

RECORRER UN ARRAY I

SENTENCIA **FOREACH**

Es un bucle utilizado para recorrer una matriz (o array)

foreach (\$array as \$valor) {}

El bucle se ejecutará tantas veces como elementos tenga la matriz

\$array es el array que queremos recorrer

\$valor es el valor de cada posición del array

RECORRER UN ARRAY II

Para recorrer un array y contar sus elementos utilizaremos el bucle **FOREACH**:

```
<?php
    $dias = ["domingo", "lunes", "martes", "miercoles", "jueves", "viernes", "sábado"];

    foreach ($dias as $dia) {
        echo "Hoy es $dia";
        echo "<br>";
    }
?>
```

```
Hoy es domingo
Hoy es lunes
Hoy es martes
Hoy es miercoles
Hoy es jueves
Hoy es viernes
Hoy es sabado
```

Para saber los elementos que tiene un array:

`count($dias)` o `sizeof($dias)`

EJERCICIO

Recorrer la siguiente lista con un bucle imprimiendo el doble de cada número:

\$lista = [1,9,3,8,5,7]

El doble de 1 es 2
El doble de 9 es 18
El doble de 3 es 6
El doble de 8 es 16
El doble de 5 es 10
El doble de 7 es 14

RECORRER UN ARRAY ASOCIATIVO

Para recorrer un array asociativo y recuperar tanto los valores como las claves:

```
foreach ($matriz as $clave => $valor) {}
```

En este caso, además del valor, recuperamos el índice del array

\$matriz es el array que queremos recorrer

\$valor es el valor de cada posición del array

\$clave es el índice del array que se está tratando en ese momento

EJERCICIO

A partir del ejercicio de los países y sus capitales, recorrer el array y mostrar la lista de éstos.

Capital de Francia es París
Capital de Ecuador es Quito
Capital de Gran Bretaña es Londres
Capital de Alemania es Berlin
Capital de Argentina es Buenos Aires

BUSCAR ELEMENTOS EN UN ARRAY I

Dado el siguiente array escalar:

```
$dias = ["lunes", "martes", "miercoles", "jueves"];
```

Para buscar un elemento concreto:

```
$clave = array_search("miercoles", $dias);
```

Si lo encuentra, la variable **clave** tendrá el valor de la posición del elemento dentro del array (en este caso 2)

Si no lo encuentra la variable **clave** valdrá **null**

Si ahora queremos borrar este elemento:

```
if ($clave !== null) { //con != el primer elemento no se borraría  
    array_splice($dias, $clave, 1); //o unset($dias[$clave]);  
}
```

BUSCAR ELEMENTOS EN UN ARRAY II

Dado el siguiente array asociativo:

```
$dias = ["Dia1"=>"lunes", "Dia2"=>"martes", "Dia3"=>"miercoles",  
"Dia4"=>"jueves", "Dia5"=>"viernes"];
```

Para buscar un elemento concreto:

```
$clave = array_search("miercoles", $dias);
```

Si lo encuentra, la variable **clave** tendrá el valor de la clave del elemento dentro del array (en este caso **Dia3**)

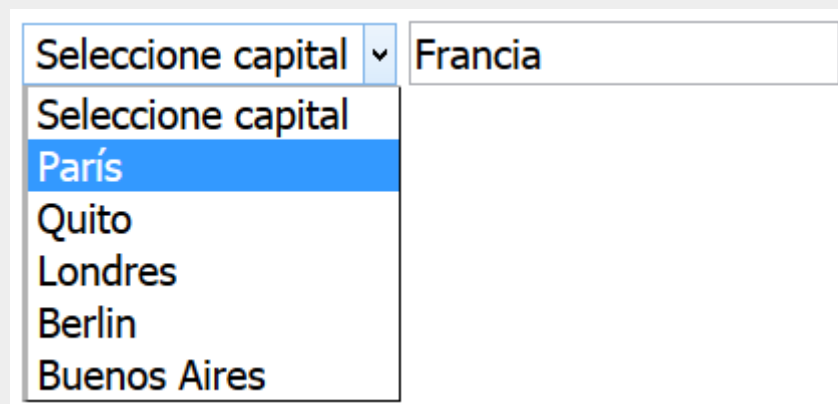
Si no lo encuentra la variable **clave** valdrá **null**

Si ahora queremos borrar este elemento:

```
if ($clave !==null) { //con != el primer elemento no se borraría  
    unset($dias[$clave]); //array_splice() no funciona con arrays  
    asociados  
}
```

EJERCICIO

A partir del ejercicio de países y capitales:



The image shows a web form with two elements. On the left is a dropdown menu with the label 'Seleccione capital'. The menu is open, showing a list of capital cities: París, Quito, Londres, Berlin, and Buenos Aires. 'París' is currently selected and highlighted in blue. To the right of the dropdown is a text input field containing the word 'Francia'.

Mediante un formulario con una combo seleccionaremos una capital, la buscaremos en el array y mostraremos el país en un campo de `input` (utilizar el evento `onchange`)

```
<select name="capital" onchange='this.form.submit()>
```

COMPROBAR ELEMENTOS DEL ARRAY

Si tenemos el siguiente array:

```
$dias = ["Dia1"=>"lunes", "Dia2"=>"martes", "Dia3"=>"miercoles"];
```

Si queremos comprobar la existencia de un elemento en un array:

```
$existe = in_array("lunes", $dias); //true o false
```

Si queremos comprobar la existencia de una clave en un array:

```
$existe = array_key_exists("Dia1", $dias); //true o false
```

ORDENAR ARRAYS I

Para ordenar arrays podemos utilizar:

sort → ordena de menor a mayor perdiendo la relación clave/valor

```
$colores = ['A'=>'rojo', 'B'=>'azul', 'C'=>'verde', 'D'=>'naranja'];  
sort($colores);
```

rsort → ordena de mayor a menor perdiendo la relación clave/valor

```
$colores = ['A'=>'rojo', 'B'=>'azul', 'C'=>'verde', 'D'=>'naranja'];  
rsort($colores);
```

```
sort() -> ordena de menor a mayor perdiendo la relacion clave/valor  
Array ( [A] => rojo [B] => azul [C] => verde [D] => naranja )  
Array ( [0] => azul [1] => naranja [2] => rojo [3] => verde )
```

```
rsort() -> ordena de mayor a menor perdiendo la relacion clave/valor  
Array ( [A] => rojo [B] => azul [C] => verde [D] => naranja )  
Array ( [0] => verde [1] => rojo [2] => naranja [3] => azul )
```

ORDENAR ARRAYS II

asort → ordena de menor a mayor conservando la relación clave/valor

```
$colores = ['A'=>'rojo', 'B'=>'azul', 'C'=>'verde', 'D'=>'naranja'];  
asort($colores);
```

arsort → ordena de mayor a menor conservando la relación clave/valor

```
$colores = ['A'=>'rojo', 'B'=>'azul', 'C'=>'verde', 'D'=>'naranja'];  
arsort($colores);
```

`asort()` -> ordena de menor a mayor conservando la relacion clave/valor

Array ([A] => rojo [B] => azul [C] => verde [D] => naranja)

Array ([B] => azul [D] => naranja [A] => rojo [C] => verde)

`arsort()` -> ordena de mayor a menor conservando la relacion clave/valor

Array ([A] => rojo [B] => azul [C] => verde [D] => naranja)

Array ([C] => verde [A] => rojo [D] => naranja [B] => azul)

ORDENAR ARRAYS III

ksort → ordena las claves de menor a mayor conservando la relación clave/valor

```
$colores = ['A'=>'rojo', 'B'=>'azul', 'C'=>'verde', 'D'=>'naranja'];  
ksort($colores);
```

krsort → ordena las claves de mayor a menor conservando la relación clave/valor

```
$colores = ['A'=>'rojo', 'B'=>'azul', 'C'=>'verde', 'D'=>'naranja'];  
krsort($colores);
```

```
ksort() -> ordena las claves de menor a mayor conservando la relacion clave/valor  
Array ( [A] => rojo [B] => azul [C] => verde [D] => naranja )  
Array ( [A] => rojo [B] => azul [C] => verde [D] => naranja )
```

```
krsort() -> ordena las claves de mayor a menor conservando la relacion clave/valor  
Array ( [A] => rojo [B] => azul [C] => verde [D] => naranja )  
Array ( [D] => naranja [C] => verde [B] => azul [A] => rojo )
```

EJERCICIO

Crear un array con 10 valores aleatorios entre 1 y 100 (*).

Ordenar el array de menor a mayor

A continuación extraer el array en una cadena de caracteres separados por punto y coma

En cada paso mostrar el contenido del array resultante y la cadena de caracteres final.

(*) Para generar números aleatorios podemos utilizar `$num=rand(1, 100);`

RECOGER RADIO BUTTONS

Podemos recoger los valores de un radio button de la siguiente forma:

```
<input type='radio' name='radio' value='radio 1'>  
<input type='radio' name='radio' value='radio 2'>
```

```
<?php  
    if (isset($_REQUEST['radio'])) {  
        $radio = $_REQUEST['radio'];  
        echo $radio;  
    }  
?>
```

CHECKBOXES

Cuando tenemos un formulario con más de un elemento del tipo checkbox podemos utilizar la siguiente estructura para recuperar el/los que están seleccionado:

```
if (isset($_POST['enviar'])) {  
    $numero=$_POST["numero"];  
    $count = count($numero);  
    for ($i = 0; $i < $count; $i++) { //o foreach ($numero as $valor)  
        echo $numero[$i].'<br>';  
    }  
}
```

```
<form action="#" method="post">  
    <input type="checkbox" name="numero[]" value="1"/> 1 <br/>  
    <input type="checkbox" name="numero[]" value="2"/> 2 <br/>  
    <input type="checkbox" name="numero[]" value="3"/> 3 <br/>  
    <input type="checkbox" name="numero[]" value="4"/> 4 <br/>  
    <input type="submit" name='enviar'>  
</form>
```

ARRAYS MULTIDIMENSIONALES

Un elemento de un array puede ser, a su vez, otro array.

```
$array = array();  
$array[0] = array('a1', 'a2', 'a3', 'a4');  
$array[1] = array('b1', 'b2', 'b3', 'b4');  
$array[2] = array('c1', 'c2', 'c3', 'c4');  
$array[3] = array('d1', 'd2', 'd3', 'd4');
```

| | 0 | 1 | 2 | 3 |
|---|----|----|----|----|
| 0 | a1 | a2 | a3 | a4 |
| 1 | b1 | b2 | b3 | b4 |
| 2 | c1 | c2 | c3 | c4 |
| 3 | d1 | d2 | d3 | d4 |

Para recorrer este array de dos dimensiones y mostrar claves y datos:

```
foreach ($array as $claveFilas => $fila) {  
    foreach ($fila as $claveColumna => $dato) {  
        echo "$claveFilas - $claveColumna --> $dato";  
    }  
}
```

ARRAYS MULTIDIMENSIONALES

También podemos declarar arrays asociativos.

```
$array = array();  
$array['4444444p'] = array('nombre' => 'Otilio Gotera', 'direccion' => 'calle  
mortadelo, 14');  
$array['4446644p'] = array('nombre' => 'Pepe Gotera', 'direccion' => 'calle  
filemon pi, 24');
```

| NIF | Nombre | Dirección |
|----------|---------------|----------------------|
| 4444444p | Otilio Gotera | calle mortadelo, 14 |
| 4446644p | Pepe Gotera | calle filemon pi, 24 |