UT3-1

PROGRAMACIÓN BASADA EN LENGUAJES DE MARCAS CON CÓDIGO EMBEBIDO

ARRAYS

1. Arrays

```
<HTML>
    <HEAD>
        <TITLE>Definición de matrices</TITLE>
    </HEAD>
    <BODY>
        <CENTER><H3>Uso del constructor array()</H3>
        <?php
            $Estad = array(1=>"Alemania", "Austria",5=> "Bélgica");
        ?>
        <TABLE BORDER="1" CELLPADDING="1" CELLSPACING="2">
            <TR ALIGN="center" >
                <TD>Elemento</TD>
                <?php
                    foreach ($Estad as $clave => $valor)
                        echo"<TD>$clave</TD>";
                ?>
            </TR>
            <TR ALIGN="center" >
                <TD>Valor</TD>
                <?php
                    foreach ($Estad as $clave => $valor)
                        echo "<TD> $valor </TD>";
                ?>
            </TR>
        </TABLE>
    </BODY>
</HTML>
```

Uso del constructor array()

Elemento	1	2	5
Valor	Alemania	Austria	Bélgica

1. Arrays

- Un array es una variable que almacena una secuencia de valores.
- Puede tener un número variable de elementos.
- Cada elemento puede tener un valor.
- Este valor puede ser simple (número, texto, etc.) o compuesto (otro array).
- Un array que contiene otro/s array se llama ***multidimensional***.
- PHP admite:
  - Array Escalares → los índices son números
  - Array Asociativos → los índices son cadenas

1. Arrays

- Sintaxis:
  - array ([clave =>] valor, ...)
  - La clave es una cadena o un entero no negativo.
  - El valor puede ser de cualquier tipo válido en PHP, incluyendo otro array
  - Ejemplos:

    $color = array ('rojo'=>101, 'verde'=>51, 'azul'=>255);

    $medidas = array (10, 25, 15);
  - Acceso:

    $color['rojo'] // No olvidar las comillas

    $medidas[0]
  - El primer elemento es el 0

1. Arrays

- Como el resto de variables, los arrays no se declaran, ni siquiera para indicar su tamaño.
- Pueden ser dispersos (se implementan como tablas hash).
  - Los índices de sus elementos no tienen porque ser consecutivos.
    - **$vec[1] = '1º elemento';**
    - **$vec[8] = '2º elemento';**
- En realidad contienen un mapeo entre *claves y valores* (*arrays* asociativos)
  - Array array([index]=>[valor], [index2]=>[valor], ...);
  - Los índices no tienen porque ser números
    - **$vec['tercero'] = '3º elemento';**

1. Arrays

- **Los arrays no son homogéneos**
  - Sus elementos pueden ser de cualquier tipo (incluso otros arrays) y ser de tipos diferentes.
    - **$vec[5] = '4º elemento';**
    - **$vec[1000] = 5.0;**

2. Creando y eliminando arrays

- Hay dos formas de crear un array:
  - **Asignación directa.**
    - Se añaden sus elementos uno a uno, indicando el índice (mediante []).
    - Si el array no existía se crea.

```
$vec[5] = '1° elemento'; $vec[1] = "2° elemento";
$vec[] = '3° elemento'; $vec[6]= "3° elem"; ..
```

  - **Utilizando el constructor array().**
    - Se añaden entre paréntesis los primeros elementos del array. El primero de todos tiene asignado el índice cero.

```
$vec = array(3, 9, 2.4, 'Juan');
  // $vec[0]=3; $vec[1]=9; $vec[2]=2.4;...
```

    - Se pueden fijar el índice con el operador =>

```
$vec = array(2=>3 ,9, 2.4, 'nombre'=>'Juan');
// $vec[2]=3; $vec[3]=9;.. $vec['nombre']= "Juan"
```

2. Creando y eliminando arrays

- Ejemplo:

```
$unarray = array("dia" => 15, 1 => "uno", "mes" => 9);
```

- Ejemplo:

```
$otro = array("unarray" => array(0=>14, 4=>15),
              "nombre" => "Una tabla");
```

- Para eliminar un elemento del array hay que emplear **unset()**
  - unset($miarray['nombre']);
  - unset($miarray);

- Los arrays no se imprimen con *echo*, sino con ***print_r***:

```
print_r ($unarray) ;
```

```
Array ( [dia] => 15 [1] => uno [mes] => 9 )
```

3. Arrays: Arrays escalares

- **Ejemplo 1:**

```
$trimestre1 = array(1 => 'Enero', 'Febrero', 'Marzo');
print_r($trimestre1);
```

Array ([1] => Enero [2] => Febrero [3] => Marzo) Array ([1] => Enero [2] => Febrero [3] => Marzo)

Genera:

```
Array (    [1] => Enero,
           [2] => Febrero,
           [3] => Marzo)
```

Array que empieza en 1 en vez de 0

3. Arrays: Arrays escalares

- **Ejemplo 2:**

```
$array = array(1,1,1,1,1, 8=>1 ,4=>1,19, 3=>13);
print_r($array);
```

Genera:

```
Array(    [0] => 1, [1] => 1,
          [2] => 1, [3] => 13,
          [4] => 1, [8] => 1,
          [9] => 19 )
```

El valor 13 sobrescribe al anterior de la posición 3.
El valor 19 se aloja en la pos 9, que es la siguiente a la última utilizada (8).

3. Arrays: Arrays escalares

- Mostrar el contenido del array (for)

```
$ciudades = array("París", "Madrid", "Londres");
for ($i=0; $i<=count($ciudades); $i++) {
    echo $ciudades[$i];
    echo "<br>";
}
```

- Mostrar el contenido del array (foreach)

```
$ciudades = array("París", "Madrid", "Londres");
foreach ($ciudades  as  $ciudad) {
    echo $ciudad;
    echo "<br>";
}
```

4. Arrays: Arrays asociativos

- La clave o índice es un String.

- Pueden definirse:
  - Mediante la función array():

```
$precios = array("Azúcar" => 1,  "Aceite" => 4,  "Arroz" => 0.5);
$capitales = array("Francia"=>"París",  "Italia"=>"Roma");
```

  - Por referencia:

```
$precios["Azúcar"] = 1;
$precios["Aceite"] = 4;
$precios["Arroz"] = 0.5;

$capitales["Francia"]="París";
$capitales ["Italia"]="Roma";
```

4. Arrays: Arrays asociativos

- Mostrar el contenido del array asociativo

```
$precios = array("Azúcar" => 1, "Aceite" => 4, "Arroz" => 0.5);
echo "<ul>";
    foreach ( $precios as $producto => $precio ) {
        echo "<li>"."Producto: ".$producto." Precio: ".$precio."</li>";
    }
echo "</ul>";
```

- Producto: Azúcar Precio: 1
- Producto: Aceite Precio: 4
- Producto: Arroz Precio: 0.5

- Otra forma (mucho menos utilizada)

```
while (list($clave,$valor) = each ($precios)) {
        echo "Producto: ".$clave."Precio: ".$valor."<br />";
}
```

5. Arrays: Arrays multidimensionales

- Son arrays en los que al menos uno de sus valores es, a su vez, otro array.

- Pueden ser escalares o asociativos

```
$pais=array(
        "espana"=>array(
                        "nombre"=>"España",
                        "lengua"=>"Castellano",
                        "moneda"=>"Peseta"),
        "francia" =>array(
                        "nombre"=>"Francia",
                        "lengua"=>"Francés",
                        "moneda"=>"Franco"));
```

Arrays *multidimensionales*

pais	idioma	moneda
España	Castellano	Peseta
Francia	Francés	Franco

5. Arrays: Arrays multidimensionales

```
001  <?php
002     $juan=array('Juan García Glez.',175,40);
003     $ana=array('Ana Madrid Llorente',162,86);
004     $alumnos=array($juan,$ana);
005     print_r( $alumnos);
006  ?>
```

```
001  <?php
002     $alumnos=array(
003        array('Juan García Glez.',175,40),
004        array('Ana Irene Palma',162,86));
005     print_r( $alumnos);
006  ?>
```

6. Arrays de dos dimensiones

- Realiza el código *php* necesario para visualizar la siguiente tabla. Utiliza un array unidimensional:

País	Capital	Extensión	Habitantes
Alemania	Berlín	557046	78420000
Austria	Viena	83849	7614000
Bélgica	Bruselas	30518	9932000

7. Arrays: Recorrer un array

- **Recorrer un array secuencial:**
  - Usar count($matriz) y un bucle
    - **int count (mixed $array)**
    - Devuelve el número de elementos que contiene el array.

```
$matriz=array('lunes','martes','miércoles','jueves','viernes','sábado',
'domingo');
echo count($matriz);
```

- Otra función para el tamaño de la matriz
  - sizeof($matriz)
    - Devuelve el número de elementos.

7. Arrays: Recorrer un array

- **Recorrer un array no secuencial o asociativo:**
  - A través de funciones que actúan sobre un puntero interno:
    - **current()** – devuelve el valor del elemento que indica el puntero
    - **pos()** – realiza la misma función que current
    - **reset()** – mueve el puntero al primer elemento del array
    - **end()** – mueve el puntero al último elemento del array
    - **next()**– mueve el puntero al elemento siguiente
    - **prev()** – mueve el puntero al elemento anterior
    - **count()** – devuelve el número de elementos de un array
    - **key()** – devuelve el índice de la posición actual.

7. Arrays: Recorrer un array

- Ejemplo:

```
$semana = array("lunes", "martes", "miércoles", "jueves",  "viernes",
            "sábado", "domingo");
echo count($semana);        //7
reset($semana);             //situamos el puntero en el 1ºelemento
echo current($semana);      //lunes
next($semana);
echo pos($semana);          //martes
end($semana);
echo pos($semana);          //domingo
prev($semana);
echo current($semana);      //sábado
echo key($semana);          // 5
```

7. Arrays: Recorrer un array

- **list($var1, $var2, $var3, ...)**
  - Asigna valor a una lista de variables en una sola operación. Solo arrays numéricos

```
list($var1, $var2)= array("Lunes", "Martes");
$var1="Lunes", $var2="Martes"
```

- **each($unarray)**
  - En cada iteración devuelve el par clave/valor actual y avanza el cursor. Devuelve un array de 4 elementos:

    0,key → la clave

    1,value → el valor

```
$semana = array("lunes", "martes");
foreach ($semana as $k=>$v) {
    echo "<pre>";
    print_r(each($semana));
    echo "</pre>";
}
```

```
Array
(
    [1] => lunes
    [value] => lunes
    [0] => 0
    [key] => 0
)

Array
(
    [1] => martes
    [value] => martes
    [0] => 1
    [key] => 1
)
```

7. Arrays: Recorrer un array

- **array_keys ($unarray [,valor_a_buscar])**
  - Devuelve las claves del array en otro array
  - Si hay *val_a_buscar,* sólo devuelve las claves de ese valor

```
$matriz = array(0=>100,  "color"=>"rojo");
print_r(array_keys($matriz));
$matriz = array("azul","red","green","azul","azul");
print_r(array_keys($matriz, "azul"));
```

```
Array
(
    [0] => 0
    [1] => color
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
```

- **array_values ($unarray)**
  - Devuelve todos los valores del array en orden numérico.

```
$matriz = array("talla" => "XL", "color" =>"dorado");
print_r(array_values($matriz));
```

```
Array
(
    [0] => XL
    [1] => dorado
)
```

8. Arrays: Buscar un elemento

- **array preg_grep(string patron, array $matriz)**
  - Devuelve un array con los elementos que cumplen el criterio fijado por *patron*.

- **array_search(valor, $matriz)**
  - Permite buscar un valor en un array y si lo encuentra devuelve su clave, si no devuelve NULL.

8. Arrays: Buscar un elemento

- **in_array(valor, $matriz, $strict)**

  - Devuelve *True* o *False* en función de la existencia o no de un valor en el array. Si *$strict* es *True se* tendrá en cuenta el tipo de los valores.

  - Es case-sensitive.

```
$a = array('1.10', 12.4, 1.13);
if (in_array('12.4', $a, true)) {
        echo "'12.4' Encontrado con chequeo STRICT\n";}
if (in_array(1.13, $a, true)) {
        echo "1.13 Encontrado con chequeo STRICT\n";}
```

8. Arrays: Buscar un elemento

- **array_count_values($matriz)**
  - Cuenta las veces que aparece cada elemento de un array en ese array

```
$matriz = array(1, "hola", 1, "mundo", "hola");
array_count_values($matriz);     // devuelve array(1=>2,"hola"=>2,"mundo"=>1)
```

9. Arrays : Modificar un array

- **mixed array_pop (array &$matriz)**

  - Extrae y devuelve el último elemento del array. Obsérvese que esta función actúa sobre el array original como indica el hecho de que reciba el **argumento** implícitamente **por referencia**.

```
$matriz=array('lunes','martes','miércoles','jueves','viernes','sábado','domingo');
echo array_pop($matriz);
var_dump($matriz);
```

9. Arrays : Modificar un array

- **int array_push(array &$matriz, $var1, $var2, ...)**
  - Inserta los elementos $var al final del array y devuelve el número de elementos que contiene el array aumentado.

```
$matriz=array('lunes','martes','miércoles','jueves','viernes','sábado');
echo (array_push($matriz,'domingo'));
var_dump($matriz);
```

9. Arrays : Modificar un array

- **mixed array_shift (array &$matriz)**
  - Extrae el primer elemento de la matriz, desplazando todos los elementos restantes hacia adelante. Devuelve el elemento extraído.

```
$matriz=array('lunes','martes','miércoles','jueves','viernes','sábado','domingo');
echo array_shift($matriz);
var_dump($matriz);
```

- **array_unshift ($mat, $elem1, $elem2, …)**
  - Permite añadir uno o más elementos por el inicio de la matriz indicada como parámetro. Devuelve el nuevo número de elementos del array.

9. Arrays : Modificar un array

- **array_walk (matriz, func_usuario [, parametro])**
  - Nos permite aplicar una función definida por el usuario a cada uno de los elementos de un array.
  - La función *func_usuario()* recibe, al menos, dos parámetros
    - El valor del elemento
    - Su clave asociada
  - Una vez aplicada la función, el puntero interno del array se encontrará al final de él.

```
function aEuros(&$valor,$clave)   {
      $valor=$valor/166.386;
}
array_walk($precios,'aEuros');
```

Producto	Precio
prod1	1500 Ptas.
prod2	1000 Ptas.
prod3	800 Ptas.
prod6	100 Ptas.
prod7	500 Ptas.

Producto	Precio
prod1	9.02 €
prod2	6.01 €
prod3	4.81 €
prod6	0.60 €
prod7	3.01 €

9. Arrays : Modificar un array

- **array array_replace (array &$matriz_destino , array &$matriz_origen)**

  - Devuelve un array que es el resultado de sobrescribir/añadir sobre matriz destino los elementos de matriz origen (los que coinciden en índice se sobrescriben, y los que no se añaden). No afecta a las matrices que recibe como argumento.

```
$matriz_destino = array('altura'=>181,'peso'=>77);
$matriz_origen = array('pelo'=>'moreno‘,'peso'=>85);
var_dump(array_replace($matriz_destino, $matriz_origen));
```

9. Arrays : Modificar un array

- **array_merge($mat1, $mat2, $mat3)**
  - Une las matrices indicadas como parámetros, empezando por la primera. Elimina los elementos con claves duplicadas (dejando la última leída).
  - También podemos unir matrices con el operador + . Elimina claves duplicadas (dejando el primer elemento leído).

9. Arrays : Modificar un array

- **array_merge_recursive($mat1,$mat2,$mat3)**
  - Permite combinar matrices sin perder elementos. Devuelve la matriz resultado de la suma. Con las claves duplicadas genera una nueva matriz para ese elemento.

- **array_pad($mat, $cantidad, $relleno)**
  - Permite añadir elementos de relleno en el inicio (negativo) y fin del array (positivo) hasta completar $cantidad. Devuelve la matriz resultado.

9. Arrays : Modificar un array

- **array array_slice (array $matriz , int $inicio, int $cantidad)**
  - Devuelve un sub-array de $matriz a partir del *inicio* indicado y con la cantidad de elementos indicada.
  - Si cantidad no se especifica devuelve todos los elementos desde *inicio* hasta el final.

```
$vec = array(10,6,7,8,23);
$res = array_slice($vec,1,3);    // $res= 6,7,8
```

Inicio	
Positivo	Posición del primer elemento contando desde el principio.
Negativo	Posición de comienzo desde el final
Cantidad	
Positivo	Número de elementos a considerar
Negativo	Se detendrá a tantos elementos del final.
Nulo	Se consideran todos los elementos hasta el final

9. Arrays : Modificar un array

- **array array_splice (array $matriz , int $inicio, int $cantidad, mixed $reemplazo)**

  - Elimina de matriz *cantidad* elementos contados a partir del parámetro *inicio*, los sustituye por los elementos del array *reemplazo* y los devuelve en un array. Si los índices son numéricos los reajusta.

```
$matriz=array('lunes','martes','miércoles','jueves','viernes','sábado‘,'domingo');
var_dump (array_splice($matriz,1,2));
var_dump($matriz);

$matriz=array('altura'=>181,'peso'=>75,'pelo'=>'moreno');
var_dump(array_splice($matriz,1,2));
var_dump($matriz);
```

9. Arrays : Modificar un array

- **string implode (string $delimitador , array $matriz)**
  - Convierte *matriz* en una cadena de caracteres separando sus elementos con la cadena indicada en *delimitador*.

```
$matriz=array(7,'julio',2011);
echo implode(' de ',$matriz);
```

9. Arrays : Modificar un array

- **Intersección de matrices.**
  - array_intersect($mat1,$mat2,$mat3)
    - Devuelve una matriz con los elementos comunes a las matrices indicadas. La comparación se hace con el operador identidad (===)
  - array_intersect_assoc($mat1,$mat2,$mat3)
    - Devuelve una matriz con los elementos comunes utilizando el operador identidad (===). En la comparación se tienen en cuenta también las claves.

9. Arrays : Modificar un array

- **Creación de una matriz con los elementos únicos de otra:**
  - **array_unique ($mat)**
    - Crea una nueva matriz a partir de otra original, tomando sólo los elementos no duplicados de ésta. Utiliza el operador de identidad en la comparación.
  - **array_combine ($mat1,$mat2)**
    - Crea un nuevo array a partir de otros dos. Un array le sirve para tomar las claves y el otro para tomar los valores correspondientes. Los dos arrays deben tener el mismo número de elementos.

9. Arrays : Modificar un array

- **array_reverse ($array, true)**
  - Devuelve el array invertido.
  - Si el 2º parámetro es true, conserva las claves

```
$entrada = array("php", 4, "rojo");
$resultado = array_reverse($entrada);
$resultado_claves = array_reverse($entrada, true);
```

```
Array
(
[0] => rojo
[1] => 4
[2] => php
)
Array
(
[2] => rojo
[1] => 4
[0] => php
)
```

- **range (low, high, paso)**
  - Crea una matriz que contiene un rango de elementos
  - *paso* indica el salto

```
$numeros = range(5,9);  (5,6,7,8,9)
$numeros2 = range(0,50,10);  (0,10,20,30,40,50)
$letras = range(a,f);  (a,b,c,d,f)
```

9. Arrays : Modificar un array

- **compact(var1,var2,….,varN)**
  - Crea un vector asociativo cuyas claves son los nombres de las variables y los valores el contenido de las mismas.

```
$ciudad = "madrid";
$edad = "21";
$vec = compact("ciudad”,"edad");

Es equivalente a:
$vec = array("ciudad"=>"madrid", "edad"=>"21");
```

- **shuffle ($array)**
  - Desordena en forma aleatoria los elementos de un array.

10. Arrays : Ordenar un array

- **Ordenación de matrices**:
  - **bool sort (array &$array [, int $sort_flags = *SORT_REGULAR*])**
    - Ordena un array de menor a mayor
  - **bool rsort (array &$array [, int $sort_flags = *SORT_REGULAR*])**
    - Ordena un array en orden inverso (de mayor a menor)
  - **bool asort (array &$array [, int $sort_flags = *SORT_REGULAR*])**
    - Ordena un array manteniendo la correlación de los índices con los elementos asociados.
  - **bool arsort (array &$array [, int $sort_flags = *SORT_REGULAR*])**
    - Ordena un array en orden inverso, manteniendo la correlación de los índices con los elementos asociados.
  - **bool ksort (array &$array [, int $sort_flags = *SORT_REGULAR*])**
    - Ordena un array por clave, manteniendo la correlación entre la clave y los datos.

10. Arrays : Ordenar un array

- **Ordenación de matrices:**

  - **bool krsort (array &$array [, int $sort_flags = *SORT_REGULAR*])**
    - Ordena un array por clave en orden inverso, manteniendo la correlación entre la clave y los datos.

  - **bool usort (array &$array , callable $value_compare_func)**
    - Ordena un array usando una función de comparación definida por el usuario. Se asignan nuevas claves a los elementos ordenados.

  - **bool uksort (array &$array , callable $key_compare_func)**
    - Ordena las claves de un array usando una función de comparación proporcionada por el usuario.

10. Arrays : Ordenar un array

- **Ordenación de matrices:**

  - **bool uasort (array &$array , callable $value_compare_func)**
    - Ordena un array de manera que los índices mantienen sus correlaciones con los elementos del array asociados, usando una función de comparación definida por el usuario.

  - **bool array_multisort (array &$arr [, mixed $arg = *SORT_ASC* [, mixed $arg = *SORT_REGULAR* [, mixed $...]]])**
    - Ordenar varios arrays al mismo tiempo, o un array multi-dimensional por una o más dimensiones. Las claves asociativas (string) se mantendrán, aunque las claves numéricas son re-indexadas.