Arreglos en PHP

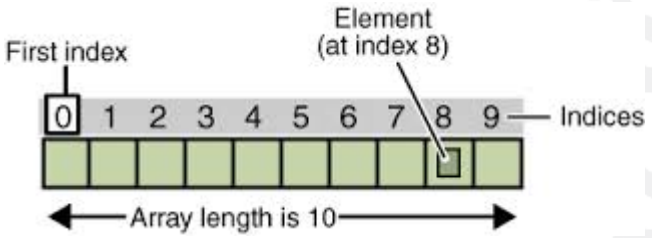
Arreglos Bidimensionales (Vectores)

Introducción

Una array (arreglo) es un conjunto de variables indexadas y que forman en una sola y sencilla súper variable que ofrece una manera fácil de pasar varios juntos. Durante gran parte de este capítulo, vamos a ver el funcionamiento interno de los arreglos y explorar todas las funciones integradas en PHP para su manipulación.

Características

- Los arreglos (o arrays en inglés) son variables que almacenan una lista de valores en forma secuencial.
- Utilizan un índice que indica la posición del dato y el dato en sí mismo en dicha posición.
- En PHP lo arreglos empiezan desde la posición 0 (y no en uno como en otros lenguajes de programación).
- Internamente PHP entiende como vector a un mapa de tipo de datos que se asocia con valores y claves.



Ejemplo Completo de Vectores

```
<?php

$nombres[] = 'Eduardo';

$nombres[] = 'Miguel';

$nombres[] = 'Ivan';

$nombres[] = 'Esteban';

echo '<p>'.$nombres[0].'</p>';

echo '<p>'.$nombres[1].'</p>';

echo '<p>'.$nombres[2].'</p>';

echo '<p>'.$nombres[3].'</p>';
```

?>

Inicializar un Array

Existen varias formas de inicializar arreglos. Como puede verse en el ejemplo anterior, los datos son cargados (o inicializados) con la forma: $vector[] = valor y automáticamente los valores se irán almacenando de manera secuencial desde la posición 0 en adelante.

También se podría hacer lo siguiente:

```
$nombres[1] = 'Eduardo';

$nombres[2] = 'Miguel';

$nombres[3] = 'Ivan';

$nombres[4] = 'Esteban';
```

Lo que sucedería es que el vector comenzaría desde la posición 1 a diferencia del ejemplo 20 donde comienza en la posición 0.

Otra forma de inicializar un vector:

```
$nombres = array('Eduardo', 'Miguel', 'Ivan', 'Esteban');
```

Verificar si una variable es Array

Al incorporar los Arrays en su aplicación, a veces se necesita saber si una determinada variable es Array. La función, is_array (), está disponible para llevar a cabo esta tarea, su estructura es:

boolean is_array(mixed variable)

Ejemplo:

```
<?php
```

```
//un array puede contener diversos tipos de dato dentro!
$test = array('Pedro', 20, true);
if ( is_array($test) ){
    echo '<p>La variable $test es un Array</p>';
}
else {
    echo '<p>La variable $test no es un Array</p>';
}

$test2 = 500;
if ( is_array($test2) ){
    echo '<p>La variable $test2 es un Array</p>';
}
else {
    echo '<p>La variable $test2 no es un Array</p>';
}
?>
```

Contar cantidad de elemento de un Vector

La función count cuenta la cantidad de elementos que tiene un arreglo.

Ejemplo de la función count combinada con un loop FOR

```
<?php

    // sin usar count, es importante entender que si el Array crece o decrece en cantidad de elementos, debo modificar la variable $cant
    $dias = array( 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo' );
    $cant = 7;
    for ( $i=0; $i <= $cant; $i++){
```

```
        echo '<p>'.$dias[$i].'</p>';
    }

    echo '<hr>'; //imprimimos un separador en pantalla

    // usando count, el contenido de $cant es dinámico
    $dias = array( 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo' );
    $cant = count($dias);
    for ( $i=0; $i <= $cant; $i++){
        echo '<p>'.$dias[$i].'</p>';
    }

?>
```

Arreglos no secuenciales

Los índices no necesariamente respetan el orden secuencia de incremento de 1 en 1.

Al cargar el arreglo se debe especificar la posición donde se almacena cada valor.

Estos arreglos no se pueden recorrer utilizando el ciclo FOR sino que hay que utilizar el ciclo FOREACH

Su principal uso es cuando el índice del vector representa un valor de utilidad en la lógica del negocio.

Ejemplo de arreglos no secuenciales

```
<?php

    //en este caso el índice es utilizado como un elemento del negocio: indica el ID de socio
    $socios[115] = 'Jorge';
    $socios[160] = 'Silvia';
```

```
    $socios[15] = 'Armando';

    $socios[20] = 'Roberto';

    $socios[725] = 'Fernanda';

    $socios[05] = 'Martina';

    $socios[117] = 'Alejandrina';


    echo '<p>El socio 115 es '.$socios[115].'</p>';

    echo '<p>El socio 20 es '.$socios[20].'</p>';


    /*

    ¿Cómo recorrer con un loop un array que no presenta una secuencia en sus indices?

    */

?>
```

Arreglos asociativos

Características

Se caracterizan por utilizar como índices palabras alfabéticas, a diferencia de los arreglos que hemos visto en los puntos anteriores que utilizan valores enteros como índices.

Ejemplo de Arreglos Asociativos

```
<?php

    $persona['nombre'] = 'Juan';

    $persona['edad'] = 18;

    $persona['estudiante'] = true;


    echo '<p>Nombre: '.$persona['nombre'].'</p>';

    echo '<p>Edad: '.$persona['nombre'].'</p>';
```

```
    if( $persona['estudiante'] ){
        echo '<p>'.$persona['nombre'].' estudia</p>';
    } else {
        echo '<p>'.$persona['nombre'].' no estudia</p>';
    }

    /*
    ¿Cómo recorrer todas las propiedades del array $persona?
    Usando el loop foreach
    */
?>
```

Operador "=>"

Podemos construir un array asociativo utilizando el operador especial =>

```
<?php
    $persona = array('nombre'=>'Juan', 'edad'=>18, 'estudiante'=>true);

    echo '<p>Nombre: '.$persona['nombre'].'</p>';
    echo '<p>Edad: '.$persona['nombre'].'</p>';

    /* el resto de mi código */
?>
```

Para mejorar la legibilidad, muchos desarrolladores se aprovechan de que el intérprete PHP no tiene en cuenta las tabulaciones y saltos de linea, entonces generan de esta forma el array

```
<?php

    $persona = array(
            'nombre'=>'Juan',
            'edad'=>18,
            'estudiante'=>true
            );

echo '<p>Nombre: '.$persona['nombre'].'</p>';
    echo '<p>Edad: '.$persona['nombre'].'</p>';

    /* el resto de mi código */

?>
```

Bucle FOREACH

Definición

FOREACH es una construcción repetitiva como el FOR o WHILE que permite iterar sobre arreglos. Especialmente más parecida al FOR, ya que repite una cantidad exacta de veces que es específicamente la longitud del Array que se desea recorrer. Solo funciona con arreglos y arrojará un error si se utiliza con otro tipo de datos.

Estructura corta

```
<?php

    foreach($array as $valor){
```

```
        /*
este bloque se ejecuta tantas veces
como elementos tenga $array
dentro de este bloque, la variable $valor
almacena el valor correspondiente a cada posición del array
*/
                }

?>
```

En cada iteración se asigna en la variable valor el contenido del elemento del arreglo donde señala el puntero interno y luego el mismo se incrementa en una unidad.

Cuando foreach comienza su primera ejecución el puntero interno del arreglo se reinicia en forma automática apuntando al primer elemento.

Estructura completa

```
<?php

    foreach($array as $indice=>$valor){
        /*
este bloque se ejecuta tantas veces
como elementos tenga $array
dentro de este bloque, la variable $indice almacena el valor correspondiente al índice de la posición (un número o String, dependiendo si es un array secuencial o asociativo)
y la variable $valor almacena el valor correspondiente a cada posición del array
*/
                }

?>
```

Ejemplo Completo de FOREACH

```
<?php

    echo '<h2>Recorriendo arrays secuenciales</h2>';
    $dias = array( 'Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo' );
    foreach($dias as $nombre){
       echo '<p>'.$nombre.'</p>';
    }

    echo '<h2>Recorriendo arrays no secuenciales</h2>';

       //en este caso el índice es utilizado como un elemento del negocio: indica el ID de socio
       $socios[115] = 'Jorge';
       $socios[160] = 'Silvia';
       $socios[15] = 'Armando';
       $socios[20] = 'Roberto';
       $socios[725] = 'Fernanda';
       $socios[05] = 'Martina';
       $socios[117] = 'Alejandrina';

    echo '<h3>Nombre de socios</h3>';
       foreach( $socios as $nombre ){
          echo '<p>'.$nombre.'</p>';
       }

       echo '<h3>Nombre y número de socios</h3>';
       foreach( $socios as $indice=>$nombre ){
```

```
    echo '<p>El socio $indice es '.$nombre.'</p>';
}

echo '<h2>Recorriendo arrays asociativos</h2>';

$persona['nombre'] = 'Juan';
$persona['edad'] = 18;
$persona['estudiante'] = true;

foreach($persona as $prop=>$val){
    //noten que al hacer echo de un boolean, sale un 1 o nada en pantalla!
    echo '<p>La propiedad '.$prop.' = '.$val;
}

?>
```

Función range

Permite crear un arreglo que tiene rangos de elementos. Llena el valor de un arreglo desde un número indicado como inicio hasta un numero indicado como final. También permite rellenar el arreglo con letras.

Ejemplo Completo de la función range()

```
<?php
```

```
    echo '<h2>Utilizando range() para generar un rango del 1 al 10</h2>';
    $numeros = range(1, 10);

    foreach($numeros as $valor){
       echo '<p>'.$valor.'</p>';
    }

    echo '<h2>Utilizando range() para generar un rango alfabetico a-z</h2>';
    $letras = range('a', 'z');

    foreach($letras as $valor){
       echo '<p>'.$valor.'</p>';
    }

?>
```

Imprimir un Array en pantalla para propósitos de prueba

Muchas veces durante el desarrollo de nuestra aplicación, se necesita saber cual es el contenido de un Array como propósito de prueba. Para estos casos se puede utilizar la función print_r(). Esta función de PHP recibe como parámetro una variable de tipo Array y nos muestra fácilmente su contenido a la pantalla para realizar pruebas:

boolean print_r(mixed variable [, boolean return])

Otra función muy utilizada es var_dump()

Ejemplo completo de print_r y var_dump

```
<?php
$capitales = array('Paris', 'Buenos Aires', 'Lima', 'Madrid', 'Montevideo', 'Brasilia');
```

```
//utilizando print_r

print_r($capitales);


//utilizando var_dump

//el beneficio de var_dump es que nos muestra el tipo de dato interpretado en ese momento

//y permite 'mostrar en pantalla' la cantidad de variables que queramos, separando por comas

//var_dump($var1, $var2, $varN);

var_dump($capitales);


//al utilizar var_dump muchos desarrolladores utilizan un <pre> para mejorar la legibilidad en pantalla

echo '<pre>';

var_dump($capitales);

echo '</pre>';

?>
```

var_dump como herramienta de trabajo

Como se puede probar en el ejercicio anterior, var_dump nos ofrece más información, lo que lo hace ideal para "entender qué está sucediendo en determinado momento de mi script". Es importante saber que var_dump puede ser utilzado con cualquier tipo de variable.

Agregar y Remover elementos de un Array

Introducción

PHP proporciona una serie de funciones tanto para agregar como para quitar elementos de un Array. Algunas de estas funciones se proporcionan como una conveniencia para los programadores que desean imitar varias implementaciones de colas (FIFO, LIFO, etc), como se refleja en sus nombres (push, pop, shift y unshift). A continuación se presentan estas funciones y varios ejemplos.

Agregar un valor al inicio de un Array

La función array_unshift () añade elementos al inicio del Array. Todas las claves numéricas preexistentes son modificadas para reflejar su nueva posición en el Array, pero los índices asociativos no se ven afectados. La sintaxis es:

int array_unshift(array array, mixed variable [, mixed variable...])

El siguiente ejemplo agrega dos capitales al Array:

```
<?php

$capitales = array('Paris', 'Buenos Aires', 'Madrid', 'Montevideo', 'Brasilia');


array_unshift($capitales, 'Lima', 'La paz');


print_r( $capitales ); // Lima, La paz, Paris, Buenos Aires, Madrid, Montevideo, Brasilia

?>
```

Agregar un valor al final de un Array

La función array_push() añade elementos al final del Array retornando la cantidad total de elementos del mismo, luego de haber agregado los valores nuevos. Esta función permite agregar múltiples elementos de una sola vez enviándolos como parámetros. El prototipo es:

int array_push(array array, mixed variable [, mixed variable...])

El siguiente ejemplo agrega dos capitales al final del Array:

```
<?php

$capitales = array('Paris', 'Buenos Aires', 'Madrid', 'Montevideo', 'Brasilia');


array_push($capitales, 'Lima', 'La paz');


print_r( $capitales ); //Paris, Buenos Aires, Madrid, Montevideo, Brasilia, Lima, La paz

?>
```

Remover un valor del inicio del Array

La función array_shift() remueve el primer elemento del Array retornando el valor removido. Si el Array es secuencial todos los valores correspondientes se desplazarán hacia abajo, mientras que en los Arrays asociativos no serán afectadas las claves. La sintaxis es:

mixed array_shift(array array)

El siguiente ejemplo remueve el primer elemento del Array:

```
<?php
$capitales = array('Paris', 'Buenos Aires', 'Madrid', 'Montevideo', 'Brasilia');

$resu = array_shift($capitales);

print_r( $capitales );        //Buenos Aires, Madrid, Montevideo, Brasilia
echo $resu;                   //Paris
}
?>
```

Remover un valor del final del Array

La función array_pop() remueve el último elemento del Array retornando el valor removido. La sintaxis es:

mixed array_pop(array array)

El siguiente ejemplo remueve el último elemento del Array:

```
<?php
$capitales = array('Paris', 'Buenos Aires', 'Madrid', 'Montevideo', 'Brasilia');

$resu = array_pop($capitales);
```

```
print_r( $capitales );        //Paris, Buenos Aires, Madrid, Montevideo

echo $resu;                   //Brasilia

}

?>
```

Localización de elementos en un Array

Búsqueda en un Array

La función in_array() busca en un Array un valor específico. Retorna TRUE si el valor se encontró y FALSE en caso contrario: La sintaxis es:

boolean in_array(mixed needle, array haystack [, boolean strict])

```
<?php

$capitales = array('Paris', 'Buenos Aires', 'Madrid', 'Montevideo', 'Brasilia');


if (  in_array('Buenos Aires', $capitales) )

{

      echo 'Valor encontrado';

}

?>
```

Búsqueda de un índice(clave) en un Array Asociativo

La función key_exists() devuelve TRUE si la clave especificada se encuentra en el Array y FALSE en caso contrario. La sintaxis es:

boolean array_key_exists(mixed key, array array)

```
<?php

$edades = array( 'Juan' => 28, 'Carlos' => 16, 'Maria' => 35, 'Raquel' => 46, 'Graciela' => 34 );


$buscar = 'Raquel';

if (array_key_exists($buscar, $edades)){

    echo 'La edad de '.$buscar.' es '.$edades[$buscar];

} else {

    echo $buscar.' no se encuentra en el listado';

}

?>
```

Búsqueda de un valor en un Array Asociativo

La función array_search() busca en un Array un valor específico. Si lo encuentra devuelve el índice (la clave) correspondiente sino devuelve FALSE. La sintaxis es:

mixed array_search(mixed needle, array haystack [, boolean strict])

```
<?php

$edades = array( 'Juan' => 28, 'Carlos' => 16, 'Maria' => 35, 'Raquel' => 46, 'Graciela' => 34 );


$buscar = 28;

$indice = array_search($buscar, $edades);

if ( $indice ){

    echo $indice.' esta asociado al valor '.$buscar;

} else {

    echo 'No encontramos ningun elemento con el valor '.$buscar;

}

?>
```

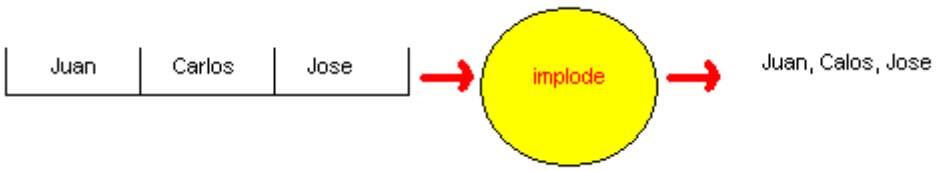
Conversión de arrays a cadena de caracteres

Función implode()

Esta función une elementos de un arreglo (vector) en una cadena de caracteres. El primer parámetro especifica cómo serán separados los diferentes elementos del vector. El segundo parámetro es el arreglo a exportar en una cadena de caracteres

```
<?php

$array = array('Juan', 'Carlos', 'Jose');
$separado_por_comas = implode(",", $array);

echo $separado_por_comas; // Juan,Carlos,Jose

?>
```

Gráficamente sucede lo siguiente:



Función explode()

Divide una cadena de caracteres en varios elementos de un Array.

Recibe tres parámetros:

- Delimitador: La cadena delimitadora.
- Cadena: Cadena de caracteres a dividir en elementos el vector
- Límite (optativo): Si el parámetro limit es positivo, el array devuelto contendrá el máximo de elementos en el limit y el último elemento contendrá el resto del string. Si el parámetro limit es negativo, se devolverán todos los componentes a excepción el último -limit. Si el parámetro limit es cero, actuará como si su valor fuera 1. Si no utilizamos el 3er parámetro, el String se divide en todos los elementos que encuentre una vez aplicado el delimitador.

```
<?php
```

```
$str = 'Juan|Carlos|Jose';

print_r(explode('|', $str));



/*

Resultado…

Array
(
   [0] => Juan
   [1] => Carlos
   [2] => Jose
)

*/
?>
```

Ordenamientos

Sin duda, el ordenamiento de los datos es un tema central de la informática. Cualquiera que haya tomado una clase de programación es muy consciente de los algoritmos de ordenamiento, como burbujeo u otros. PHP facilita el proceso por el que ofrece una multitud de funciones útiles capaces de ordenar Arrays en una variedad de maneras.

Función “array_reverse()”

Esta función invierte el orden de los elementos de un Array.

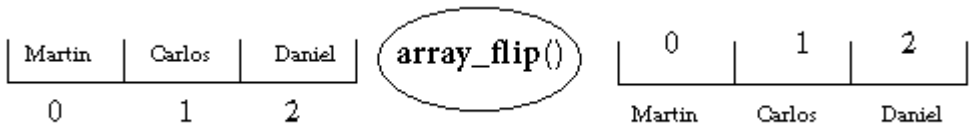
Función “array_flip()”

La función array_flip() invierte los roles de las claves (índices del vector) y sus valores correspondientes.

```
<?php

$state = array("Martin", "Carlos", "Daniel");

$state = array_flip($state);

print_r($state);

?>
```

El resultado será:

Array ([Martin] => 0 [Carlos] => 1 [Daniel] => 2)



Ordenamientos en Arreglos Secuenciales

Función “sort()”

Ordena los valores de un arreglo por orden alfabético.

Ejemplo Completo de la función sort()

```
<?php
    $personas[11] = 'Ivan';
    $personas[16] = 'Miguel';
    $personas[15] = 'Ricardo';
    $personas[20] = 'Fabian';
    $personas[40] = 'Lorena';
    $personas[33] = 'Veronica';
    $personas[17] = 'Eduardo';

    sort($personas);
    foreach($personas as $pos=>$nombre){
        echo '<p>'.$nombre.' tiene la posicion '.$pos.'</p>';
    }

    /*
    Resultado...
    <p>Eduardo tiene la posicion 0</p>
```

```
        <p>Fabian tiene la posicion 1</p>
        <p>Ivan tiene la posicion 2</p>
        <p>Lorena tiene la posicion 3</p>
        <p>Miguel tiene la posicion 4</p>
        <p>Ricardo tiene la posicion 5</p>
        <p>Veronica tiene la posicion 6</p>

        Como ven, la función sort redefine los indices de un array
        */
?>
```

Función “rsort()”

Ordena los valores de un arreglo por orden inverso.

Importante

Tanto sort como rsort redefinen los indices del array, esto puede traernos problemas si utilizamos las posiciones por ejemplo para identificar IDs de usuarios.

Ordenamientos en Arreglos Asociativos

Función “asort()”

Ordena un arreglo asociativo y mantiene la asociación de índices.

Función “arsort()”

Ordena un arreglo asociativo en forma descendiente (reversa) y mantiene la asociación de índices.

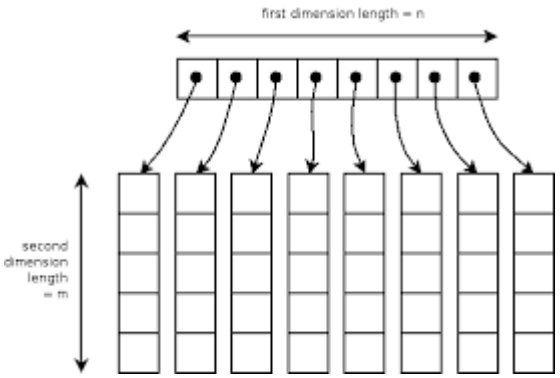
Otra forma de cargar un arreglo asociativo:

```
$edades = array('Juan' => 22, 'Martin' => 18, 'Ignacio' => 21);
```

Arreglos Multidimensionales (Matrices)

Introducción

Hasta ahora, los arreglos de ejemplos que hemos visto han sido de una dimensión, con sólo un nivel de claves entre corchetes. Sin embargo, PHP puede soportar fácilmente múltiples arreglos unidimensionales con un número arbitrario de las llaves.



Definición

Al igual que con los vectores unidimensionales, no hay necesidad de declarar nuestras intenciones de antemano. La primera referencia a una variable de matriz puede ser una asignación como:

$multi_array[1][2][3][4][5] = 'dato almacenado';

En realidad, en nuestra opinión, el pensamiento de las matrices como multidimensional es más confusa de lo que necesita. En su lugar, sólo se debe recordar que los valores que están almacenados en arrays pueden ser matrices, como legítimamente como pueden ser cadenas o números. El índice de múltiples sintaxis en el ejemplo anterior es simplemente una forma concisa para referirse a un (cuatro dimensiones) de matriz que se almacena con una llave de 1 en $multi_array. Se debe tener en cuenta también que puede tener distintas profundidades de referencia en diferentes partes de la matriz, así:

```
$multi_level_array[0] = 'un simple string';

$multi_level_array[1]['contains'] = 'un string almacenado mas profundo';
```

Características

Son arreglos que contienen arreglos.

Una matriz es un arreglo de dos dimensiones. Esto significa que se necesita especificar con dos índices para acceder al valor de un elemento.

Ejemplo Completo de Arreglos Multidimencionales

```
<?php

    $alumnos_aula0 =array('Juan' => 22, 'Martin' => 18, 'Ignacio' => 21);
    $alumnos_aula1 =array('Carlos' => 52, 'Roberto' => 55, 'Horacio' => 49);
    $todos = array( $alumnos_aula0, $alumnos_aula1 );

    foreach($todos as $indice=>$alumnos){
        echo '<p>Imprimiendo alumnos del aula '.$indice.'<p>';
        echo '<ul>';
        //como cada elemento del array $todos A SU VEZ es un array, puedo recorrerlo con otro foreach!
        foreach($alumnos as $nombre=>$edad){
            echo '<li>'.$nombre.' ('.$edad.')</li>';
        }
        echo '</ul>';
    }

    /*
    Resultado...
    <p>Imprimiendo alumnos del aula 0<p>
<ul>
    <li>Juan (22)</li>
    <li>Martin (18)</li>
    <li>Ignacio (21)</li>
</ul>
<p>Imprimiendo alumnos del aula 1<p>
```

```
    <ul>
        <li>Carlos (52)</li>
        <li>Roberto (55)</li>
        <li>Horacio (49)</li>
    </ul>
      */


?>
```

Resumen

Un arreglo es un tipo de variable, con la singularidad de que no contiene un único valor, sino un conjunto de valores referenciados con un índice. La sintaxis es muy parecida a la usada con el resto de variables, con diferencias en la forma en que el arreglo es creado y lógicamente, la forma en que su valor es recuperado (pues teniendo un valor múltiple, tenemos que indicar qué valor, o en qué forma, deseamos que nos sea devuelto).

Los arrays organizan sus elementos utilizando indices. Si los indices son numéricos y progresivos, el array es secuencial. Si los indices son numéricos no progresivos es un array no secuencial. Los índices pueden ser cadenas de caracteres (Strings), conformando un array asociativo.

Foreach es una estructura de repetición (loop) diseñada para trabajar con arrays secuenciales, no secuenciales y asociativos. No deberíamos utilizar for o while si lo que queremos hacer es recorrer todos los elementos de un array.

También se pueden usar cadenas de texto como índices. Este tipo es el arreglo asociativo.

Un arreglo multidimensional es un arreglo en el que al menos uno de sus valores es, a su vez, un arreglo.