Programación para Internet

Módulo 4. Programación en ambientes Web del lado del servidor

Objetivo: Aplicar los conceptos básicos de la programación estructurada utilizando un lenguaje empotrado en el servidor Web.

4.1. Introducción
4.2. Identificadores y tipos de datos
4.3. Control del Flujo
4.4. Manejo de cadenas y expresiones regulares
4.5. Arreglos
4.6. Manejo de archivos y directorios
4.7. Programación con formularios
4.8. Validación de formularios

La versión más reciente de PHP es la 5.3.8 (23 de agosto 2011), que incluye todas las ventajas que provee el nuevo Zend Engine 2.

PHP

PHP Hypertext Pre-processor

PHP es un lenguaje de scripting ampliamente utilizado para fines generales que es especialmente adecuado para el desarrollo web y puede ser embebido en páginas HTML.

Generalmente se ejecuta en un servidor web, tomando el código en PHP como su entrada y creando páginas web como salida.

Ejemplo

```
<?php
        $a = 'Hola Mundo';
        echo $a;
?>
```

PHP Tags

Un bloque de código PHP está incrustado dentro de código HTML con las siguientes etiquetas.

1-<?php ... ?>

2-<script language="php"> ... </script>

3-<? ... ?>

4-<% ... %>

Anatomía del Script

- Todas las sentencias como llamadas a funciones, asignaciones de variables, salidas de datos, directivas y demás deben terminar con punto y coma (;).
- Los bloques de código se envuelven entre llaves ({}).

Comentarios

Pueden ser incluidos en el código utilizando varios estilos:

```
// Comentario de una línea

# Este es otro estilo de comentario

/* Así es como
 puedo crear
  un comentario de varias líneas */
```

Comentarios...

```
/**
* API Documentation Example
*
* @param string $bar
*/
function foo($bar) { }
```

Tipos de datos

- Escalares
  - boolean
  - int
  - float
  - String
- Compuestos
  - array
  - object

Valores numéricos

- Octales se representan con un 0 al inicio
- Hexadecimal se representa con 0x
- Exponenciales con E, 2E7

Cuidado con los números

```
echo (int) ((0.1 + 0.7) * 10);
```

- Esperamos que la expresión ((0.1 + 0.7) * 10) devuelva 8, sin embargo, imprime 7.
- Esto pasa porque el resulta de esta expresión aritmética es guardada internamente como 7.999999 en vez de 8

Cadenas

- Secuencia de caracteres
- PHP puede crear cadenas con comillas o apóstrofes

```
print 'Esto funciona';

print "Esto tambien";
```

Caracter de escape

- Se usa la diagonal inversa
- \t \n \r

```
print "Esta cadena tiene \": comillas!";

print 'Esta cadena tiene \': apostrofe!';
```

Otro caso

```
print "Esta cadena tiene ': apostrofe!";

print 'Esta cadena tiene ": comillas!';
```

Curiosidades

```
$var = 'Esta es una cadena
       con dos lineas';

$query = "SELECT max(order_id)
          FROM orders
          WHERE cust_id = $custID";
```

Variables en cadenas entrecomilladas

```
$number = 45;
$vehicle = "camion";
$message = "Este $vehicle tiene
capacidad de  $number personas";
```

// Muestra "Este camion tiene capacidad de 45 personas

```
print $message;
```

Cuidado con las variables

```
$memory = 256;

$message = "My computer has
$memoryMbytes of RAM";

$message = "My computer has
{$memory}Mbytes of RAM";
```

Conversión de tipos de datos

```
$x = 10.88;
echo (int) $x;

$y = settype($x,integer);
```

Conversiones automáticas

- $var = "100" + 15; //$var = 115
- $var = "100" + 15.0; //$var = 115.0
- $var = 39 . " Steps"; // $var = “39 Steps”
- $var = 39 + " Steps"; //$var = 39
- $var = "3 blind mice" + 39; //$var = 42
- $var = "test" * 4 + 3.14159; //$var = 3.14159

Funciones para tipos de datos

- boolean is_int(mixed variable)
- boolean is_float(mixed variable)
- boolean is_bool(mixed variable)
- boolean is_string(mixed variable)
- boolean is_array(mixed variable)
- boolean is_object(mixed variable)
- string gettype(mixed expression)
- var_dump(mixed expression [, mixed expression ...])

Variable

- En PHP todas las variables comienzan con el símbolo de pesos ($) y no es necesario definir una variable antes de usarla.
- Tampoco tienen tipos, es decir que una misma variable puede contener un número y luego puede contener caracteres.
- Son variables case-sensitive.

Ejemplo

```
$a=1;
$a=3.34;
$a="Hola Mundo";
$var;
$VAR;
```

Variable Variables

```
$name = 'foo';
$$name = 'bar';
echo $foo;      // Muestra 'bar'
```

- Comenzamos por crear una variable que contiene la cadena foo. A continuación, utilizamos una sintaxis especial $$ para indicar que queremos que el intérprete utilice el contenido de $name para hacer referencia a una nueva variable, por lo tanto se crea la nueva variable $foo.

Variable variables...

```
$name = '123';
// 123 es un identificador no válido

$$name = '456';

echo ${'123'};
// Usamos llaves para tomar el
identificador
```

Variable variables

```
function myFunc() {
    echo 'myFunc!';
}

$f = 'myFunc';
$f(); // ejecutará myFunc();
```

isset()

- Funcion que devuelve true si una variable dada existe

```
echo isset ($x);
```

empty()

- Función que evalua si una variable es igual a *false*

```
echo empty($x)
```

unset()

- Una variable puede ser destrozada con la función unset()

```
unset($var);
```

Constantes

- Variable que no cambia su valor durante la ejecución

```
define("CONSTANT", "Hello world.");
```

Salidas de datos

- PRINT Y ECHO:

- Pueden enviar cualquier tipo de combinación de cadenas estáticas, números, matrices y otros tipos de variables
- Básicamente desempeñan el mismo trabajo.

Tanto print como echo por lo general son vistos con paréntesis:

```
echo "hola";

echo ("hola");
```

Los paréntesis no hacen ninguna diferencia con el comportamiento de impresión.

Sin embargo, cuando se utilizan con echo, sólo un parámetro de salida se puede proporcionar.

Diferencias

- PRINT es una función, de manera que devuelve verdadero o falso.
- Como PRINT es una función, se vuelve mas lento que ECHO.
- ECHO acepta varios parámetros concatenando con coma (,) o punto (.).

// imprime "The answer is 42”

```
echo "The answer is ", 42;
```

Diferencias

- Usar comas (,) en ECHO es mas rápido que usar puntos (.).
- Ambos permiten utilizar apóstrofes (') o comillas (“).
- Recuerda que es más rápido el proceso con apóstrofes (').
- ECHO son menos letras a escribir que PRINT.

Die

- Die(), es un alias de exit().
- Permite terminar un script y mandar una cadena de salida o regresar un numero al proceso desde el cual se mando a llamar el script.

Operadores

- Los operadores nos sirven para realizar operaciones entre variables, por ejemplo, sumas, restas, comparaciones, etc, la sintaxis es bastante natural con algunas excepciones.

Operador Aritmético

- Suma (+)
- Resta (-)
- Multiplicación (*)
- División (/)
- Módulo (%)
- Incrementar (++)
- Decrementar (--)

Operadores de Asignación

- Asignación (=)
- Unión o concatenación (.)

Operadores De Comparación

- Igual (==)
- Igual y mismo tipo (===)
- Diferente (!=) o (<>)
- Diferente el valor y el tipo (!==)
- Menor que (<)
- Mayor que (>)
- Menor o igual que (<=)
- Mayor o igual que (>=)

Operadores Lógicos

- AND (And) o (&&)
- OR (Or)
- XOR (Xor)
  - Una u Otra pero no Ambas
- NOT (!)

Operadores para bits

- &
- |
- ^
- <<
- >>

Punteros

```
$a = 10;
$b = &$a; // puntero
$b = 20;
echo $a; // imprime 20
```

Operador para suprimir errores

- Previene la muestra de mensajes de errores que las funciones puedan tener

```
$x = @mysql_connect();
```

Condicionales

- Las sentencias condicionales nos permiten ejecutar o no unas ciertas instrucciones dependiendo del resultado de evaluar una condición.
- Las más frecuentes son la instrucción IF y la instrucción SWITCH.

Sentencia If

```
<?php
   if (condición)
   {
      Sentencias a ejecutar
   }
   else
   {
      Sentencias a ejecutar
   }
?>
```

If ... elseif

```
if ($var < 5)
    print "El valor es pequeño";
elseif ($var < 10)
    print "El valor es medio";
```

Casos especiales

- Si requiere comparar cadenas, utilice la función de PHP strcmp().

Sentencia Switch

```
switch ($menu)
{
    case 1: print "Uno";
            break;
    case 2: print "Dos";
            break;
    case 3: print "Tres";
            break;
    case 4: print "Cuatro";
            break;
    default:print "Otra opcion";
}
```

Otro switch

```
$score = "Distinction";
switch ($score)
{
    case "High Distinction":
    case "Distinction":
        print "Good student";
        break;
    case "Credit":
    case "Pass":
        print "Average student";
        break;
    default:
        print "Poor student";
}
```

Bucles

- Los bucles nos permiten iterar conjuntos de instrucciones, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición.

Sentencia while

```
<?php
   $i = 0;
   while ($i < 10) {
      echo $i . PHP_EOL;
      $i++;
   }
?>
```

Sentencia do-while

```
<?php
   $i = 0;
   do{
      echo $i . PHP_EOL;
      i++;
   }while ($i < 10);
?>
```

Sentencia for

```
<?php
for (inicial ; condición ; ejecutar en iteración)
{
   intrucciones a ejecutar.
}
?>
```

Tronar ciclos

- Break;
- Continue;

Manejo de errores

- PHP tiene unas excelentes facilidades para tratar con los errores, las cuales proporcionan un excelente nivel de control sobre cómo los errores se generan, se manejan y son reportados.

Tipos de errores

- Compile-time errors

Errores detectados por el parser mientras esta compilando un script. No pueden ser atrapados en otro momento.

- Fatal errors

Errores que cuelgan la ejecución de un script. No pueden ser atrapados.

- Recoverable errors

Errores que representan fallas significantes, pero que pueden ser manejados de una manera segura

Tipos de errores...

- Warnings

Errores recuperables que indican una falla en tiempo de ejecución. No cuelgan la ejecución.

- Notices

Indican que ocurrió una condición de error, pero no es necesariamente significante. No cuelgan la ejecución.

Configuración para el manejo de errores

- Afortunadamente, la mayoría de las directivas de configuración se encuentran en el php.ini.
- Las mas importantes son error_reporting, display_errors and log_errors.

set_error_handler

- Sus scripts pueden declarar un catch-all que será llamada por PHP cuando un error ocurre llamando a la función set_error_handler().
- Mediante la función set_error_handler se permite cambiar el manejador de errores por defecto de PHP por el nuestro propio.

```
$oldErrorHandler = '';

function myErrorHandler ($errNo, $errStr,
$errFile, $errLine, $errContext) {

   logToFile("Error $errStr in $errFile at
line $errLine");

   // Call the old error handler

   if ($oldErrorHandler) {

      $oldErrorHandler ($errNo, $errStr,
$errFile, $errLine, $errContext);

   }

}

$oldErrorHandler = set_error_handler
($oldErrorHandler);
```

PHP5 Exceptions

- Esta versión del lenguaje viene con la posibilidad de utilizar bloques try/catch para el control de errores.
- Permite también mediante la función set_exception_handler crear nuestro propia función de respuesta para aquellas situaciones en las que una excepción llegue al nivel principal de ejecución, sin haber sido capturada.
- ESTE TEMA SERÁ VISTO DURANTE POO

Repaso

Question

Cuáles son las dos salidas a pantalla usadas en PHP?

Answer

`echo Y print`

Question

Cuál es la diferencia entre las comillas y las apóstrofes en una cadena?

Answer

Con comillas, las variables dentro son entendidas. Con apóstrofes no es así.

Question

Todas las variables de PHP inician con cuál caracter?

Answer

Question

Cuál diferencia existe entre `if`… `elseif` y `switch`?

Answer

El método `switch` es usualmente mas compacto, facil de escribir y entender.

Question

Cuál es el propósito de usar `break` dentro de un `switch`?

Answer

Prevenir la ejecución de los siguientes casos existentes en el `switch` y brincar a la siguiente llave de cerrado

Question

Por qué a veces los programadores no usan `break` despues de un `case` específico dentro del `switch`?

Answer

Agrupar casos (2 o mas que requieren la misma ejecución)

Question

Cuál es la diferencia entre "==" y "=”?

Answer

“==” usado para expresiones de comparación y “=” es usado para asignación

Question

Cuál es la diferencia entre un **break** y un **continue** dentro de un ciclo?

Answer

Break romperá el ciclo.
Continue previene la ejecución del código y pasa a la siguiente iteración

Question

Cuál es la diferencia entre "==" y "==="?

Answer

“===” usado para comparar tipo de dato y valor “==” solo compara valor

Question

Como indicas la inicialización y cierre de script PHP?

Answer

```
<?php
?>


<script language="PHP">
</script>


<?
?>
```

Question

En que posición del archivo se debe escribir el código de PHP?

Answer

Donde sea

Question

Qué extensión debo utilizar para los códigos de PHP?

Answer

.php

Question

Cómo se ponen los comentarios?

Answer

```
// This is a one-line comment

# This is another one-line comment style

/* This is how you
   can create a multi-line
   comment */
```

Question

Cuáles son los tipos de datos básicos en PHP?

Answer

- boolean
- float
- integer
- string

Question

Cuáles son los dos tipos de datos complejos en PHP?

Answer

- array
- object

Question

Se puede cambiar el valor de una constante en PHP?

Answer

no

Question

Cómo se comparan cadenas en PHP?

Answer

con strcmp()

Question

Las variables en PHP son case sensitive?

Answer

Si

Question

Cuál es la principal diferencia entre un `while` y un `do… while` ?

Answer

- Un `while` puede que nunca se ejecute.
- Un `do…while` se ejecutará al menos una vez.

Question

Describe que es una variable variable

Answer

Es una variable que recibe el nombre de otra variable

Question

Cómo se indican los punteros?

Answer

```
$puntero = &$var;
```

Question

Cúal es el método para cambiar el manejador de errores por uno propio?

Answer

set_error_handler()

Question

Para que sirve die()

Answer

Para terminar la ejecución y mostrar un mensaje

Funciones

- El uso de funciones nos da la capacidad de agrupar varias instrucciones bajo un solo nombre y poder llamarlas a estas varias veces desde diferentes sitios, ahorrándonos la necesidad de escribirlas de nuevo.

```
function name() { }
```

Retorno de valores

```
function hello()
{
    return "Hello World";
}
$txt = hello();
echo hello();
```

Retorno de valores

- Naturalmente return también permite la interrupción de la ejecución de una función y su salida aunque no se desee devolver un valor

```
function hello($who)
{
  echo "Hello $who";
  if ($who == "World") {
    return;
}
```

Retorno de valores

- Las funciones también se pueden declarar de manera que retornen por referencia, lo que le permite devolver una variable como el resultado de la función, en lugar de una copia (las funciones devuelven una copia del valor para cada tipo de dato, excepto los objetos). Normalmente, es usado para cosas como recursos (como las conexiones de base de datos)

Alcance de variables

- Pertenecen al área en la que fueron declaradas:
  - Globales
  - Funciones
  - Clase

Paso de argumentos

```
function hello($who)
{
   echo "Hello $who";
}
hello("World");
```

Argumentos opcionales

```
function hello($who = "World")
{
    echo "Hello $who";
}
hello();
```

Distinta cantidad de argumentos

```
function hello()
{
  if (func_num_args() > 0) {
  $arg = func_get_arg(0);
  echo "Hello $arg";
  } else {
    echo "Hello World";
  }
}
hello("Reader");
hello();
```

Cantidad de argumentos variable

- func_num_args()

  Devuelve el número de argumentos

- func_get_arg()

  Devuelve un argumento en una posición

- func_get_args()

  Devuelve un arreglo con los argumentos

Argumentos por referencia

```
function countAll(&$count)
{
    $count = ..... ;
}
$count = 0;
countAll($count,
```

Archivos externos

- include
- require
- include_once
- require_once

```
<?php
 include("func1.php");
 require("func2.php");
?>
```

Arreglos

- Los arreglos son los reyes indiscutibles de las estructuras de datos avanzadas en PHP.
- Los arreglos en PHP son extremadamente flexibles, permiten llaves numéricas de incremento automático, llaves alfanuméricas o una mezcla de ambos, y son capaces de almacenar casi cualquier valor, incluyendo otros arreglos.
- Con más de setenta funciones para la manipulación de ellos, los arreglos pueden hacer en la práctica cualquier cosa que puedas imaginar-y algo más.

Arreglos...

- Todos los arreglos son colecciones ordenadas de elementos, llamados elementos.
- Cada elemento tiene un valor, y es identificada por una **llave** que es exclusiva del arreglo al que pertenece.
- Las claves pueden ser números enteros o cadenas.

Creación de arreglos

- $a = array (10, 20, 30);
- $b = array (’a’ => 10, ’b’ => 20, ’cee’ => 30);
- $c = array (5 => 1, 3 => 2, 1 => 3,);
- $d = array();
- $e = array(Hola, Adios);
- $f = array(“uno”, “dos”, “tres”);

Acceso a los valores

Los valores contenidos en un arreglo se pueden recuperar y modificar usando corchetes [ ].

- Siguiente llave numerica mas grande

```
$x[] = 10;
```

- Asignamos en que llave guardar

```
$x['aa'] = 11;
```

- Imprimimos lo guardado en la llave 0, lo cual será 10

```
echo $x[0];
```

```
$newArray[0] = "PHP";
$newArray[1] = "MySQL";
$newArray[2] = "Apache";
```

- ¿Qué hace?

```
$newArray[2] = "Cherokee";
```

```
$shopping = array( );
$shopping[] = "Leche";
$shopping[] = "Cafe";
$shopping[] = "Azucar";
```

- En que posiciones ha quedado cada elemento?

```
$array = array(
    "first"=>1,
    "second"=>2,
    "third"=>3);
```

- Qué imprime?

```
print $array["second"];
```

```
$numbers = array(
    1=>"one",
    "two",
    "three",
    "four");
```

- Cuál es la llave del item “three”??

```
$oddNumbers = array(
   1=>"one",
   3=>"three",
   5=>"five");
$oddNumbers[]="other";
```

- En que llave quedó el elemento “other”?

```
$a = array (
   '4' => 5,
   'a' => 'b');
$a[] = 44;
```

- En qué llave quedará el item 44?

```
$a = array (
   'A' => "Esta es una A",
   'B' => "Esta es una B");
$a[] = "Esta que será";
```

- En qué llave quedará el item “Esta que será”?

```
$numeros = array(1=>"uno", 3=>"tres",
5=>"cinco");

$numeros[2] = "dos";
$numeros[4] = "cuatro";
$numeros[6] = "seis";
```

- En que orden ha quedado el arreglo?

- Se dice pues que las llaves son automáticamente incrementales y siempre los elementos se agregarán en el siguiente valor mayor posible y al final del arreglo.
- Tenga en cuenta que las llaves diferencian entre mayúsculas y minúsculas. Por lo tanto, la llave "A" es diferente de 'a', pero las llaves '1' y 1 son los mismos.

Impresión de arreglos

- PHP provee dos funciones para imprimir el valor de una variable recursivamente:
- print_r()
- var_dump()

print_r() y var_dump()

- var_dump() muestra además el tipo de dato
- var_dump() puede mostrar el valor de varias variables al mismo tiempo
- print_r() puede regresar el valor como una cadena

Arreglos multidimensionales

- A menudo los datos no pueden ser representados en un arreglo sencillo de escalar en valores enteros, cadenas, booleanos, y flotantes. Algunos datos sólo pueden estar representados cuando los arreglos tienen otros arreglos de valores.

```
$array = array();
$array[] = array(
   'foo',
   'bar');
$array[] = array(
   'baz',
   'bat');
echo $array[0][1] . $array[1][0];
```

- Qué imprime?

```
$planets = array(
    array("Mercury", 0.39, 0.38),
    array("Venus",   0.72, 0.95),
    array("Earth",   1.0,  1.0),
    array("Mars",    1.52, 0.53) );

print $planets[2][0];
```

- Qué imprime?

```
$planets2 = array(
  "Mercury"=> array("dist"=>0.39, "dia"=>0.38),
  "Venus"  => array("dist"=>0.72, "dia"=>0.95),
  "Earth"  => array("dist"=>1.0,  "dia"=>1.0,
                    "moons"=>array("Moon")),
  "Mars"   => array("dist"=>0.39, "dia"=>0.53,
                    "moons"=>array("Phobos", "Deimos"))
);

print $planets2["Earth"]["moons"][0];
```

- Qué imprime?

Asignando a variables independientes

- A veces es más fácil trabajar con los valores de un arreglo mediante la asignación a las variables individuales. Si bien esto puede ser logrado mediante la extracción de elementos individuales y la asignación de cada uno de ellos a una variable diferente, PHP ofrece un acceso rápido.

```
$info = array('coffee', 'brown', 'caffeine');
```

- Todas las variables

```
list($drink, $color, $power) = $info;
echo "$drink is $color and $power makes it special.\n";
```

- Solo algunas variables

```
list($drink, , $power) = $info;
echo "$drink has $power.\n";
```

- Solo la tercera

```
list( , , $power) = $info;
echo "I need $power!\n";
```

Unión de arreglos

- Se logra con el operador +
- Los elementos con llaves repetidas solo se muestran una vez, si el contenido es distinto, esto representa perdida de información

```
$a = array (1, 2, 3);
$b = array ('a' => 1, 'b' => 2, 'c'
=> 3);
$c = $a+$b;
```

- Cuál es el contenido de $c?

```
$a = array (1, 2, 3);
$b = array ('a' => 1, 2, 3);
$c = $a+$b;
```

- Cuál es el contenido de $c?

```
$a = array ('a' => 1,
            'b' => 2,
            'c' => 3);
$b = array ('a' => 4,
            'b' => 5,
            'c' => 6);
$c = $a+$b;
```

- Cuál es el contenido de $c?

```
$clothing = array("silk", "satin",
"cotton", "rags");
$dwelling = array("house", "tree",
"palace");
$cointoss = array("heads", "tails");


$added = $cointoss + $dwelling +
$clothing;
```

- Cuál es el contenido de $added?

Unión de arreglos sin perder elementos

- La funcion **array_merge()** concatena uno o mas arreglos, agregandolos al final.
- Si los arreglos tienen llaves de cadenas iguales, entonces el ultimo valor va a sobrescribir por la vez previa en que aparecio la llave.

```
$array1 = array("color" => "red", 2,
4);
$array2 = array("a", "b", "color" =>
"green", "shape" => "trapezoid", 4);
$result = array_merge($array1,
$array2);
```

- ([color] => green, [0] => 2, [1] => 4, [2] => a, [3] => b, [shape] => trapezoid, [4] => 4)

Comparando arreglos

- $a = array (1, 2, 3);
- $b = array (1 => 2, 2 => 3, 0 => 1);
- $c = array (’a’ => 1, ’b’ => 2, ’c’ => 3);
-
- $a == $b
- $a === $b
- $a == $c

Operadores de comparación de arreglos

- ==

Mismo número de elementos con las mismas llaves sin importar el orden.

- ===

Mismo par llave/valor en el mismo orden

Tamaño de un arreglo

- El tamaño de un arreglo puede ser obtenido con la función ***count()***.
- Funciona con cualquier variable, no necesariamente arreglos, si se quiere asegurar de que lo este usando en un arreglo, deberá primero utilizar la función ***is_array()***

```
$a = array (1, 2, 4);
$b = array();
$c = 10;
echo count($a);
echo count($b);

if is_array($c)
  echo count($c);
```

Repeticiones de un elemento en el arreglo

- **count()** devuelve el total de elementos en un arreglo, pero si se desea contar de manera unica los elementos se utiliza **array_count_values()**.
- Devuelve un arreglo asociando cada elemento con la cantidad de repeticiones.

```
$pets = array(
    "Beth"=>"Dog",
    "Arabella"=>"Rabbit",
    "Meg"=>"Cat",
    "Louise"=>"Chicken",
    "Ben"=>"Dog",
    "Neda"=>"Cat");


$petFrequency = array_count_values($pets);
```

```
[Dog] => 2
[Rabbit] => 1
[Cat] => 2
[Chicken] => 1
```

Determinar si un elemento en una posición dada existe

```
$a = array ('a' => 1, 'b' => 2);
echo isset ($a['a']);
echo isset ($a['c']);

$a = array ('a' => NULL, 'b' => 2);
echo isset ($a['a']);
```

- Qué impresiones se obtienen?

Determinar si una llave existe

```
$a = array ('a' => NULL, 'b' => 2);
echo array_key_exists ($a['a']);
```

- Qué aparece en pantalla?

Buscar si un elemento existe en el arreglo

```
$a = array ('a' => NULL, 'b' => 2);
echo in_array (2, $a);
```

- Qué aparece en pantalla?

- Se puede pasar un tercer parámetro para obligar el tipo de dato.

Obtener la posición de un elemento en el arreglo

- **array_search()** funciona igual que in_array() solo que este devolverá la llave. Si el valor no fue encontrado devuelve *false*.
- Tiene un tercer parámetro opcional para obligar la concordancia del tipo de dato

```
$a = array ('a' => NULL, 'b' => 2);
echo array_search (2, $a);
```

- Qué aparece en pantalla?

Un arreglo con las llaves

- Con **array_keys()** se obtiene un arreglo con las llaves de otro arreglo.
- Se pueden poner parámetros opcionales y entonces solo devolverá las llaves de los elementos indicados en los parámetros.

```
$pets = array(
    "Beth"=>"Dog",
    "Arabella"=>"Rabbit",
    "Meg"=>"Cat",
    "Louise"=>"Chicken",
    "Ben"=>"Dog",
    "Neda"=>"Cat");


$keys = array_keys($pets);
("Beth","Arabella","Meg","Louise","Ben","Neda","Cat")


$keys = array_keys($pets,"Dog");
("Beth","Ben")
```

Un arreglo sin llaves

- La función **array_values()** genera un nuevo arreglo solamente con los valores y olvida las llaves

Eliminación de elementos

- Un elemento puede ser eliminado de un arreglo, o un arreglo completo se puede eliminar llamando a ***unset ( )***.
- Sin embargo, la eliminación de un elemento no reasigna indices.

```
$favoritos = array("PHP", "Ace",
"COBOL", "Java", "C++");
```

- Eliminar COBOL del arreglo

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

Eliminando elementos repetidos

- La función **array_unique()** devuelve un arreglo sin items repetidos.
- Si el arreglo tiene llaves, se quedará con la primer llave, y las siguientes ocasiones en que se repita, serán omitidas.

Un arreglo rellenado

- La función **array_fill()** crea un arreglo rellenado con el dato indicado.
- Se debe indicar la primer llave, la cantidad de repeticiones y el dato a repetir.

```
$unity = array_fill(2, 5, "one");
```

```
Array ( [2] => one [3] => one [4] => one [5] => one [6] => one )
```

Arreglo con una serie

- La función **array_range()** crea un arreglo con una serie dentro.
- Se debe indicar el primer valor y el último valor. Puede indicarse como parámetro opcional el tipo de incremento

```
$fifthLetters = range("a", "z", 5);
```

```
Array ( [0] => a [1] => f [2] => k
[3] => p [4] => u [5] => z )
```

Invertir un llaves por valores

- La función **array_flip()** invierte las llaves por los valores en un arreglo.

```
$a = array ('a', 'b', 'c');
array_flip ($a));
```

```
["a"]=> 0 , ["b"]=> 1 , ["c"]=> 2)
```

Invertir el orden de un arreglo

- La función **array_reverse()** invierte el orden de los elementos de un arreglo.

```
$count = array("zero", "one", "two",
"three", "four");
$countdown = array_reverse($count);
```

```
Array ( [4] => four [3] => three [2]
=> two [1] => one [0] => zero )
```

Iteraciones en un arreglo

- Existen varias funciones para poder trabajar co las iteraciones dentro de un arreglo
  - reset() Reinicia el puntero del arreglo para poder recorrerlo.
  - prev() Recorre el puntero a la posición anterior
  - next() Recorre el puntero a la siguiente posición
  - current() Entrega el valor guardado en la posición actual
  - key() Devuelve la llave de la posición actual
  - end() Mueve el puntero a la posición final

```
$array = array('foo' => 'bar', 'baz',
'bat' => 2);

function displayArray($array) {
  reset($array);
  while (key($array) !== null) {
    echo key($array).": ".current($array);
    next($array);
  }
}
```

foreach()

- Es una manera sencilla de recorrer una arreglo.

```
$array = array('foo', 'bar', 'baz');
foreach ($array as $key => $value) {
    echo "$key: $value";
}
```

Iteraciones con funciones

- array_walk()

Aplica una función definida por el usuario a cada miembro de un arreglo.

- array_walk_recursive()

Aplica una función definida por el usuario **recursivamente** a cada miembro de un arreglo.

```
$fruits = array("d" => "lemon", "a"
=> "orange", "b" => "banana", "c" =>
"apple");

function test_print($item2, $key)
{
    echo "$key. $item2<br />\n";
}
array_walk($fruits, 'test_print');
```

```
$fruits = array("d" => "lemon", "a"
=> "orange", "b" => "banana", "c" =>
"apple");


function test_alter(&$item1, $key,
$prefix)
{
    $item1 = "$prefix: $item1";
}
array_walk($fruits,
'test_print','texto');
```

```
$sweet = array('a' => 'apple', 'b' =>
'banana');
$fruits = array('sweet' => $sweet,
'sour' => 'lemon');

function test_print($item, $key)
{
    echo "$key holds $item\n";
}

array_walk_recursive($fruits,
'test_print');
```

Ordenar un arreglo

- Los elementos de un arreglo se ordenan ascedente o descendentemente, sin embargo, las llaves se perderan en el proceso.
  - **sort()** Ascendentemente
  - **rsort()** Descendentemente
- Se tiene un segundo parámetro opcional para indicar si el ordenamiento será alfabético o numérico

Banderas de ordenamiento

- SORT_REGULAR – Compara items normalmente (No cambia los tipos)
- SORT_NUMERIC – Compara items numéricamente
- SORT_STRING – Compara items como cadenas
- SORT_LOCALE_STRING – Compara items como cadenas basado en la configuración local

```
$numbers = array(24, 19, 3, 16, 56, 8, 171);
sort($numbers);
```

- ([0] => 3, [1] => 8, [2] => 16, [3] => 19, [4] => 24, [5] => 56, [6] => 171)

```
rsort($numbers);
```

- ([0] => 171, [1] => 56, [2] => 24, [3] => 19, [4] => 16, [5] => 8, [6] => 3)

```
$arr = array ("1", 10, "Maria",
"maria", "Memo", 1, 5, 9);
sort($arr);
```

- En que orden quedará este arreglo?

Ordenar un arreglo manteniendo las llaves

- Para que al ordenar las llaves se mantengan, se utilizan las funciones:
  - **asort()** Ascendentemente
  - **arsort()** Descendentemente

Ordenar un arreglo por sus llaves

- Hemos visto como ordenar un arreglo en base a sus elementos, pero también podemos ordenarlo en base a sus llaves.
  - **ksort()** Ascendentemente
  - **krsort()** Descendentemente

Ordenar un arreglo utilizando una función definida por el usuario

- Si deseas ordenar un arreglo de alguna otra manera que no sea numérica o alfabéticamente, puedes crear tu propia función y luego mandar a ordenar con esa función.
  - **usort( )** Ordena el arreglo basado en el valor de cada elemento aplicando una nueva llave
  - **uasort( )** Mantiene el par llave/valor
  - **uksort( )** Reordena los elementos basado en la llave de cada elemento

```
function cmp_length($a, $b)
{
    if (strlen($a) < strlen($b))
        return -1;
    if (strlen($a) > strlen($b))
        return 1;

    //Si el tamaño es igual
    return 0;
}
```

```
$animals = array("cow", "ox",
"hippopotamus", "platypus");

usort($animals, "cmp_length");
```

- ([0] => ox, [1] => cow, [2] => platypus, [3] => hippopotamus)

Desordenando un arreglo

- La función **shuffle()** desacomoda un arreglo de manera random. Las llaves se pierden

Obtener un elemento random

- La función **array_rand()** toma uno o mas elementos random del arreglo y devuelve la llave de estos.

Arreglos como pilas y colas

- PHP ofrece funciones para las actividades de push y pull en las pilas y para shift y unshift en las colas.
  - array_push
  - array_pop
  - array_shift
  - array_unshift

```
$stack = array();
array_push($stack, 'bar', 'baz');
$last_in = array_pop($stack);
```

```
$queue = array('qux', 'bar', 'baz');
$first_element = array_shift($queue);
array_unshift($queue, 'foo');
```

Diferencia entre dos arreglos

- La función **array_diff()** compara dos arreglos y devuelve la diferencia.
- Devuelve un arreglo con todas las entradas del arreglo1 que no estan en los demás arreglos.
- No toma en cuenta las llaves, de modo que si se desea tomar en cuenta las llaves, se utiliza **array_diff_assoc()**

Diferencia entre dos arreglos en base a sus llaves

- Para buscar las diferencias entre dos arreglos tomando en cuenta las llaves se utiliza la función **array_diff_key()**
- En ambas funciones se puede utilizar una función definida por el usuario y las funciones a utilizar son **array_diff_uassoc()** y **array_diff_ukey()**

Intersección de dos arreglos

- La función **array_intersect()** devuelve todos los elementos del arreglo1 que existen en los demás arreglos.
- Si se desea revisar también las llaves se utiliza la función **array_intersect_assoc()**.
- **array_intersect_key()** se utiliza para validar las llaves.
- **array_intersect_ukey()** y **array_intersect_uassoc()** para validar con una función
-

Repaso de arreglos

- print_r
- var_dump
- list
- is_array
- count
- array_count_values
- isset
- array_key_exists
- in_array
- array_search
- array_keys
- array_values
- unset
- array_unique
- array_fill
- array_range
- array_flip
- array_reverse
- array_walk
- array_walk_recursive

Repaso de arreglos

- reset
- prev
- next
- current
- key
- end
- sort
- rsort
- asort
- arsort
- ksort
- krsort
- usort
- uasort
- uksort
- shuffle
- array_rand
- array_merge
- array_push
- array_pop

Repaso de arreglos

- array_shift
- array_unshift
- array_diff
- array_diff_assoc
- array_diff_key
- array_diff_uassoc
- array_diff_ukey
- array_intersect
- array_intersect_assoc
- array_intersect_key
- array_intersect_uassoc
- array_intersect_ukey

Repaso

Question

¿Cómo se declaran arreglos?

Answer

```
$variable = array();

$var[]= “cosa”;
```

Question

¿Cuáles son los dos tipos de arreglos y cuáles son sus diferencias?

Answer

Normales y Asociativos

Los normales son los que no llevan un indice en cambio los asociativos si llevan indice.

Question

¿Cómo elimino un arreglo?

Answer

unset ($arreglo[pos]); este elimina una posición y no reordena.

unset ($arreglo); este elimina todo

Question

¿Cómo separo una cadena en varias partes de un arreglo?

Answer

Con la función *explode()*

$variable = explode(" ", "ahora es el tiempo");

Question

¿Cómo genero una cadena a partir de un arreglo?

Answer

Con la función implode o join

```
print "Animales que he visto: " .
implode(", ", $animalesVistos);
```

Question

¿Cómo ordeno un arreglo sin perder el orden de las llaves(índices)?

Answer

Con la función *asort()*

Question

¿Cómo ordeno de manera inversa en base al índice?

Answer

Con la función *krsort()*

Question

¿Cuál es la diferencia entre la función *array_search()* y *in_array()*?

Answer

La función *array_search()* devuelve la posición, si no existe devuelve falso.

La función *in_array()* solo dice si existe o no o devueve resultado booleano.

Question

¿Cuáles son las maneras de concatenar arreglos?

Answer

Con + y merge

La diferencia de estos son la manera de preservar la información en caso de estar repetida

Question

¿Cómo invierto un arreglo?

Answer

Con la función *array_reverse()*