Curso de Programación en PHP

Nivel I

Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología - Oro Verde - v2.1

Capítulo 1: Conceptos básicos

Arreglos

Funciones

Manejo de cadenas (Strings)

Clase 2: Arreglos, funciones y Strings

¿Qué es un Arreglo?

Un arreglo puede definirse como un grupo o una colección finita, homogénea y ordenada de elementos.

Tipos de Arreglos

- De una dimensión (**Unidimensionales**, también conocidos como Vectores)
- De dos dimensiones (**Bidimensionales**, también denominados Matrices)
- De tres o más dimensiones (**Multidimensionales**)

Clase 2: Arreglos, funciones y Strings

Arreglos

- Se pueden crear haciendo uso de la palabra reservada array().
- La sintaxis general es de la forma:

```
$arreglo = array ('clave'=>valor, 'clave'=>valor);
```

- La clave puede ser un valor numérico no negativo o una cadena de texto.
- El valor puede ser cualquier valor soportado por PHP, incluso array().
- El primer elemento es el 0.

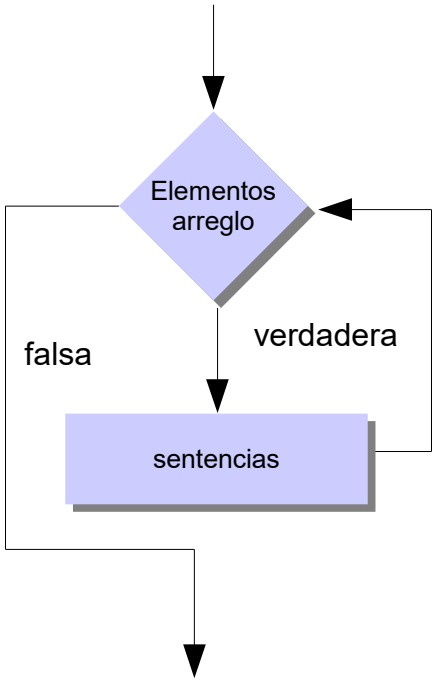
- Ejemplos:

```
$dias = array ('Domingo','Lunes','Martes','Miércoles','Jueves','Viernes','Sábado');
$caracteristicas = array ('parana'=>343,'nogoya'=>3435,'tala'=>3445);
```

Estructuras de Control - Arreglos

Estructura Selectiva **foreach**

```
foreach  (arreglo as unelemento){
        sentencias
}
```



Estructuras de Control - Ejemplo

```
$colores = array('Azul','Verde','Rojo');

foreach ($colores as $uncolor){
        print ("Color ".$uncolor);
        print ("<br>");
}
```

Resultado:

Color Azul
Color Verde
Color Rojo

print_r()

Es una función que nos permite mostrar información (contenido y estructura) sobre una variable (principalmente de arreglos y objetos) en una forma que es legible por humanos.

```
<?php

    $frutas = array ('a'=>'Anana', 'b'=>'Banana', 'm'=>'Manzana');
    print_r ($frutas);


?>
```

Salida del script:

```
Array(
    [a] => Anana
    [b] => Banana
    [m] => Manzana
)
```

Clase 2: Arreglos, funciones y Strings

sort()

Permite ordenar el arreglo que le pasamos como parámetro, modificando al mismo a nivel de claves. Esto quiere decir que destruirá todas las claves que contenga y generará uno nuevo numerando los elementos a partir del 0.

```
<?php
    $arreglo = array ('a' => 'aab', 'b' => 'bbb', 'c' => 'aaa');
    sort($arreglo);
    print_r($arreglo);
?>
```

Salida del script:
Array ([0] => aaa [1] => aab [2] => bbb)

asort()

Realiza la misma tarea que sort(), permitiendo mantener la asociación de las claves que contenga el arreglo.

```
<?php
    $arreglo = array ('a' => 'aab', 'b' => 'bbb', 'c' => 'aaa');
    asort($arreglo);
    print_r($arreglo);
?>
```

Salida del script:
Array ([c] => aaa [a] => aab [b] => bbb)

rsort() y arsort()

Estas funciones devuelven el mismos resultado que sort() y asort() pero permiten ordenar los elementos de forma descendente.

```
<?php
    $arreglo = array ('a' => 'aab', 'b' => 'bbb', 'c' => 'aaa');
    rsort($arreglo);
    print_r($arreglo);
?>
```

Salida del script:
Array ([0] => bbb [1] => aab [2] => aaa)

ksort()

Es una función que nos permite ordenar un arreglo teniendo en cuenta las claves del mismo.

```
<?php
    $arreglo = array ('d' => 'aab', 'c' => 'bbb', 'a' => 'aaa');
    ksort($arreglo);
    print_r($arreglo);
?>
```

Salida del script:
Array ([a] => aaa [c] => bbb [d] => aab)

array_diff()

Esta función calcula la diferencia entre matrices.

```
<?php

    $arreglo1 = array ('a' => 'azul', 'b' => 'verde', 'c' => 'rojo');
    $arreglo2 = array ('a' => 'amarillo', 'b' => 'naranja', 'c' => 'rojo');

    $resultado = array_diff($arreglo1, $arreglo2);
    print_r($resultado);

?>
```

Salida del script:

```
Array(
    [a ] => azul
    [b] => verde
)
```

array_intersect()

Es una función que calcula la intersección de arreglos.

```
<?php

    $arreglo1 = array ('a' => 'azul', 'b' => 'verde', 'c' => 'rojo');
    $arreglo2 = array ('a' => 'amarillo', 'b' => 'naranja', 'c' => 'rojo');

    $resultado = array_intersect($arreglo1, $arreglo2);
    print_r($resultado);

?>
```

Salida del script:

```
Array(
    [a ] => rojo
)
```

Clase 2: Arreglos, funciones y Strings

Funciones

Definición de una función

```
function nombre(parámetro1, parámetro2){
        sentencias
    return valor
}
```

donde

- nombre: nombre con el que se llamará a la función. No debe llevar espacios, acentos ni caracteres especiales.
- parámetros: son variables y/o constantes que se intercambian entre partes del código y que se tratarán como variables locales dentro de la función.
- return: establece el valor a devolver por la función.

Funciones - Ejemplo

```
function sumo($valor1, $valor2){
        $suma = $valor1 + $valor2;
    return $suma
}
```

llamada

```
$a = 5;
$b = 3;
$suma = sumo($a,$b);
print ("La suma de ".$a." mas ".$b." es: ".$suma);
```

Resultado

La suma de 5 mas 3 es: 8

Funciones – pase de parámetros

El pase de parámetros puede ser:

- por valor: la función recibe únicamente los valores. Los cambios en los mismos que se produzcan dentro de la función no se reflejan fuera de esta.
- por referencia: la función recibe una referencia al valor de una variable. Los cambios en éstos valores se verán reflejan fuera de la función.

Para que los valores sean pasados por referencia, se le debe anteponer el ampersamd(&) antes de cada parámetro en la definición de la función

Funciones – Ejemplos

Pasaje de parámetros por valor:

```
function cambioTexto($valor){
      $valor =”por más que cambie el Texto aquí, no modifico
     nada fuera de la función.”;
}

$valor=”Este es el Texto inicial de la variable.”;
cambioTexto($valor);
print ($valor);
```

Resultado

Este es el Texto inicial de la variable.

Funciones – Ejemplos

Pasaje de parámetros por referencia:

```
function cambioTexto(&$valor){
    $valor =”Ahora sí modifico el Texto fuera de la función.”;
}

$valor=”Este es el Texto inicial de la variable.”;
cambioTexto($valor);
print ($valor);
```

Resultado

Ahora sí modifico el Texto fuera de la función.

Cadenas (Strings)

Un valor string es una serie de caracteres. En PHP, un carácter es lo mismo que un byte, es decir, hay exactamente 256 tipos de caracteres diferentes.

Esto implica también que PHP no tiene soporte nativo de Unicode. Vea utf8_encode() y utf8_decode() para conocer sobre el soporte Unicode.

Nota: El que una cadena se haga muy grande no es un problema. PHP no impone límite práctico alguno sobre el tamaño de las cadenas, así que no hay ninguna razón para preocuparse sobre las cadenas largas.

Cadenas (Strings)

Las cadenas en PHP se definen de cuatro formas:

Comillas Simples → **echo** '**Ejemplo de Cadena Simple\n**'

Ejemplo de Cadena Simple\n

Comillas Dobles → **echo** “Ejemplo de Cadena Doble\n”

Ejemplo de Cadena Doble

Cadenas (Strings)

Las cadenas en PHP se definen de cuatro formas:

Heredoc →

```
$cadena = <<<FIN
Ejemplo de una cadena
usando la sintaxis heredoc.
FIN;

Ejemplo de una cadena usando la sintaxis
heredoc.
```

Nowdoc →

```
$cadena = <<<'FIN'
Ejemplo de una cadena
usando la sintaxis nowdoc.
FIN;

Ejemplo de una cadena usando la sintaxis
nowdoc.
```

Secuencias de escape en Cadenas (Strings)

Las cadenas en PHP pueden necesitar caracteres de control y para utilizarlos debemos emplear secuencias de escape utilizando la barra invertida.

En nuestro siguiente ejemplo veremos como mostrar en pantalla una comilla doble:

Ejemplo:

```
echo ” Acá quiero insertar una \“comilla doble\”. ”;
```

Salida del script:

Acá quiero insertar una “comilla doble”.

Clase 2: Arreglos, funciones y Strings

echo

Es una construcción del lenguaje PHP que trabaja de forma similar a una función pero que no requiere el uso de paréntesis. El propósito de echo es mostrar en pantalla una cadena de texto pasada como parámetro.

```
<?php
    echo “Esta es una cadena de texto<br>”;
?>
```

print()

Es una función que realiza la misma tarea que echo, permitiendo mostrar una cadena de texto por pantalla que recibe como parámetro

```
<?php

    // Cadena de texto que sale por pantalla con la función print()
    print(“Esta es otra cadena de texto”);
?>
```

printf()

Es otra función que permite mostrar en pantalla una cadena que se recibe como parámetro pero que además permite darle formato a la misma.

```
<?php

    // Se muestra una cadena por pantalla utilizando printf()
    $cadena = 10;
    printf("Hace %d horas que estoy con lo mismo.", $cadena);
?>
```

sprintf()

Trabaja de forma idéntica a printf() pero con la diferencia de que sprintf() no muestro los resultados por pantalla sino que se almacenarán en una variable.

```
<?php

    $cadena = 10;
    $cadenaResultante = sprintf("Hace %d horas que estoy con lo mismo.", $cadena);
?>
```

strtoupper()

Es una función que nos permite convertir una cadena de texto que se recibe como parámetro a mayúsculas.

```
<?php
   // Se muestra una cadena en mayúsculas por pantalla utilizando strtoupper()
   $cadena = “Cadena de texto”;
   echo strtoupper($cadena);
?>
```

Salida del script:
CADENA DE TEXTO

strtolower()

Trabaja de forma inversa a strtoupper() permitiendo convertir una cadena de texto que se recibe como parámetro a minúsculas.

```
<?php

   // Se muestra una cadena en minúsculas por pantalla utilizando strtolower()
   $cadena = “Cadena de texto”;
   echo strtolower($cadena);
?>
```

Salida del script:
cadena de texto

Clase 2: Arreglos, funciones y Strings

ucfirst()

Es una función que nos permite convertir el primer carácter de una cadena de texto a mayúsculas.

```
<?php
   // Se muestra una cadena con el primer carácter en mayúsculas
   $cadena = “cadena de texto”;
   echo ucfirst($cadena);
?>
```

Salida del script:
Cadena de texto

ucwords()

Función que nos permite convertir a mayúsculas el primer carácter de cada palabra de una cadena de texto.

```
<?php

   // Se muestra una cadena con el primer carácter en mayúsculas de cada palabra
   $cadena = “cadena de texto”;
   echo ucwords($cadena);
?>
```

Salida del script:
Cadena De Texto

trim()

Es una función que elimina los espacios en blanco al inicio y al final de una cadena que se recibe como parámetro.

```
<?php

    // Se eliminan los espacio en blanco de una cadena con la función trim()
    $cadena = "\n\t   Cadena de texto    \n     ";
    echo trim($cadena);
?>
```

ltrim()

Trabaja de forma parecida a trim() permitiendo eliminar los espacios en blanco que se encuentran al comienzo de una cadena de texto que se pasa como parámetro.

```
<?php

    // Se eliminan los espacios en blanco que se encuentran al principio de la cadena
    $cadena = "\n\t   Cadena de texto";
    echo ltrim($cadena);
?>
```

rtrim()

Trabaja de forma parecida a trim() permitiendo eliminar los espacios en blanco que se encuentran al final de una cadena de texto que se pasa como parámetro.

```
<?php

    // Se eliminan los espacios en blanco que se encuentran al final de la cadena
    $cadena = “Cadena de texto    \n”;
    echo rtrim($cadena);
?>
```

chop()

Trabaja de forma idéntica a rtrim() permitiendo eliminar los espacios en blanco que se encuentran al final de una cadena de texto que se pasa como parámetro. chop() es un alias de rtrim().

```
<?php

    $cadena = “\n\t   Cadena de texto”;
    echo chop($cadena);
?>
```

strlen()

La función strlen() nos brinda la posibilidad de obtener la longitud de una cadena de texto que se le pasa como parámetro a la misma.

```
<?php

    // Se muestra la longitud de la cadena de texto por pantalla con la función strlen().
    $cadena = “Cadena de texto”;
    echo strlen($cadena);
?>
```

count()

Esta función trabaja de forma parecida a strlen() permitiendo obtener la longitud de una cadena de texto, de arreglos y objetos.

```
<?php

    $cadena = “Cadena de texto”;
    echo count($cadena);
?>
```

Clase 2: Arreglos, funciones y Strings

substr ($texto, $inicio, $longitud)

La función substr() nos brinda la posibilidad de obtener una porción de texto de una cadena que se le pasa como parámetro a la misma.

```
<?php

    $cadena = “Cadena de texto”;
    echo substr($cadena, 1, 6);

?>
```

Salida del script:
adena

substr_replace($texto, $textoNuevo, $inicio, $longitud)

Esta función trabaja de forma parecida a substr() permitiendo encontrar y reemplazar una porción de texto de una cadena que se pasa como parámetro

```
<?php

    $cadena = “Vieja cadena de texto”;
    echo substr_replace($cadena, 'Nueva ', 0,5);

?>
```

Salida del script:
Nueva cadena de texo

str_replace(*$cadenaBuscar, $nuevaCadena, $texto*)

Esta función nos devuelve un string del cual se reemplazaron las distintas coincidencias que se encontraron. El primer parámetro de la función nos permite indicar la subcadena a buscar, el segundo parámetro nos permite indicar la subacadena a reemplazar y como tercer y último parámetro nos queda indicar la cadena de texto tratar.

```
<?php

    $cadena = “Cadena de texto”;
    $subcadena = “texto”;

    echo “Cadena: $cadena<br>”;
    echo “Subcadena: $subcadena<br>”;

    echo str_replace($subcadena, “texto (String)”, $cadena);

?>
```

Salida del script:
Cadena: Cadena de texto
Subcadena: Texto
Cadena de texto (String)

str_ireplace(*$cadenaBuscar, $nuevaCadena, $texto*)

Esta función trabaja de igual manera que str_replace() devolviendo un string del cual se reemplazaron las distintas coincidencias que se encontraron, el primer parámetro de la función nos permite indicar la subcadena a buscar, el segundo parámetro nos permite indicar la subacadena a reemplazar y como tercer y último parámetro nos queda indicar la cadena de texto tratar. **La principal diferencia** de esta función es que **no es sensitiva a mayúsculas y minúsculas.**

```
<?php

    $cadena = “Cadena de texto”;
    $subcadena = “Texto”;

    echo “Cadena: $cadena<br>”;
    echo “Subcadena: $subcadena<br>”;

    echo str_ireplace($subcadena, “texto (String)”, $cadena);

?>
```

Salida del script:
Cadena: Cadena de texto
Subcadena: Texto
Cadena de texto (String)

explode()

Permite separar (dividir) una cadena de texto, indicándole el carácter *separador* como parámetro, y nos devuelve un un arreglo con el resultado.

```
<?php

    $cadena = “2010-01-01”;
    $separador = “-”;

    echo “Cadena: $cadena<br>”;
    echo “Separador: $separador<br>”;

    $arregloAMD = explode($separador, $cadena);

    echo $arregloAMD[0].”<br>”;
    echo $arregloAMD[1].”<br>”;
    echo $arregloAMD[2].”<br>”;

?>
```

Salida del script:
Cadena: 2010-01-01
Separador: -

2010
01
01

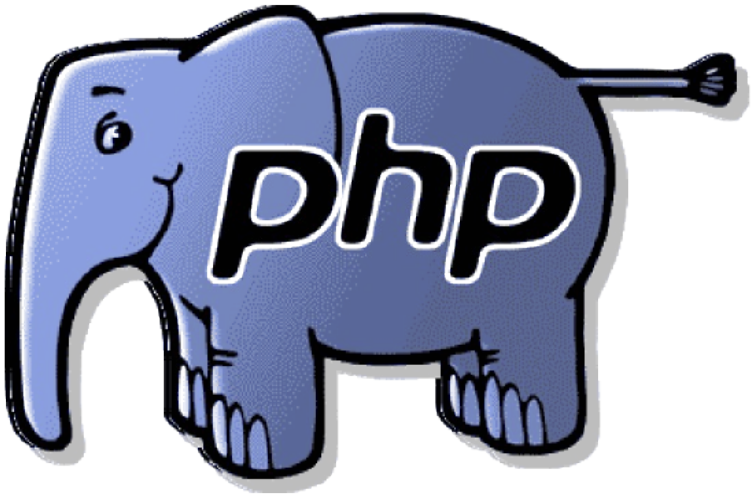
implode()

La función implode() permite juntar los elementos de un arreglo en una cadena de texto. De alguna manera trabaja de forma inversa a explode().

```
<?php

    $arreglo[0] = “2010”;
    $arreglo[1] = “01”;
    $arreglo[2] = “01”;

    $separador = “/”;

    echo implode($separador, $arreglo);

?>
```

Salida del script:
2010/01/01

¿Dudas?



¿Consultas?

Información de contacto

Web:

http://www.gugler.com.ar

http://campusvirtual.gugler.com.ar

http://www.facebook.com/gugler.com.ar

http://www.twitter.com/cgugler

Mail:

contacto@gugler.com.ar

academica@gugler.com.ar

administracion@gugler.com.ar

Versión 2.0