Índice

3. Programación de código embebido en linguaxes de marcas

3.2 Cadeas e arrays

Mentres unha **cadea** representa unha **secuencia de carácteres**, un **array** representa un **conxunto de elementos cuxo tipo non ten que ser**

carácter. Estudaranse ambos tipos a continuación debido á súa importancia e frecuente presenza nos códigos fonte PHP.

3.2.1. Cadeas

Unha **cadea**, **ou string**, **é unha serie de carácteres**. Pódense especificar mediante **comiñas simples ou comiñas dobres**. Cando se especifican con **comiñas simples** os **únicos carácteres** de **escape serán** \\ (representa o literal \) e \’ (representa o literal ‘). A continuación móstranse exemplos de cadeas.

Exemplo de cadeas en PHP

```
<?php
echo 'Isto é unha cadea sinxela';

echo 'Tamén pode incluír novas liñas en
cadeas desta forma xa que é
correcto facelo así';

// Resultado: Arnold unha vez dixo: "I'll be back"
echo 'Arnold unha vez dixo: "I\'ll be back"';

// Resultado: Borrou C:\*.*?
echo 'Borrou C:\\*.*?';

// Resultado:Borrou C:\*.*?
echo 'Borrou C:\*.*?';

// Resultado: As variables $tampouco se $expandiran
echo 'As variables $tampouco se $expandiran';
?>
```

O **único operador** de **string** en PHP a **concatenación**. Represéntase cun . e se usa para unir dous cadeas. Pódense concatenar cadeas con código HTML, por exemplo ca etiqueta
 para que se amosen fragmentos da cadea en distintas liñas. A continuación móstrase un exemplo de concatenación.

Exemplo de concatenación de dúas cadeas

```
<?php
$txt1="Ola mundo!";
$txt2="Que día tan bo!";
echo $txt1 . " " . $txt2;
// Resultado: Ola mundo! Que dia tan bo!
```

```
echo "<br/>".$txt1."<br/>".$txt2;
/*Resultado: Ola mundo!
             Que día tan bo!*/
?>
```

Soe ser útil a función **strlen()** que **devolve o número de carácteres dunha cadea**. No seguinte exemplo móstrase o seu uso.

Exemplo de uso da función strlen()

```
<?php
echo strlen("Ola mundo!");
// Resultado: 10
?>
```

Existen máis funcións nas librerías de PHP para o manexo de strings. Iranse introducindo en posteriores actividades na medida en que fagan falla.

Caracteres de escape

Caracter	Significado
\n	avance de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulador horizontal (HT o 0x09 (9) en ASCII)
\v	tabulador vertical (VT o 0x0B (11) en ASCII) (desde PHP 5.2.5)
\e	escape (ESC o 0x1B (27) en ASCII) (desde PHP 5.4.0)
\f	avance de página (FF o 0x0C (12) en ASCII) (desde PHP 5.2.5)
\\	barra invertida
\$	signo del dólar
\"	comillas dobles
\'	Comilla simple
\{$var}	Imprimir las llaves alrededor del contenido de $var

```
$a = "Hola Mundo";
$b = "Yo digo \{$a}";
echo $b; // salida: Yo digo {Hola Mundo}
```

Exemplo de uso de varias funcións sobre cadeas

```
<?php //Exemplos de funcións sobre cadeas.
        $cadea = "Estamos aprendendo PHP";
        echo "<br>Cadea orixinal: $cadea";

        echo "<br> Pasar a maiúsculas: ".strtoupper($cadea);
        echo "<br> Pasar a minúsculas: ".strtolower($cadea);
        echo "<br> Lonxitude:". strlen($cadea);
        echo "<br> Encontrar primeira ocorrencia dun carácter: ". strpos($cadea, 'e'); // ou "e"
        echo "<br> Encontrar primeira ocorrencia dun carácter (sin tener en cuenta mayús. o minús.: ". stripos($cadea, 'E'); // ou "E"
        echo "<br> Encontrar primeira ocorrencia dunha subcadea: ". strpos($cadea, 'ap'); // ou "a"
        echo "<br> Reemprazar: ". str_replace('PHP', "JAVA", $cadea);
```

```
 echo "<br> Reemprazar: ". str_replace('o', 'a', "roto");

$cadea="  Hola que tal estas  ";
echo "<br> Eliminar espazos dereita:". rtrim($cadea);
echo "<br> Eliminar espazos esquerda:". ltrim($cadea);
echo "<br> Eliminar espazos esquerda e dereita:". trim($cadea).".";

$cadea=trim($cadea);
echo "<br> Obter un trozo da cadea:". substr($cadea,0,4); //partiendo de 0 coge 4 caracteres
echo "<br> Obter un trozo da cadea:". substr($cadea,5,7); //partiendo de 5 coger 7 caracteres
echo "<br> Obter un trozo da cadea:". substr($cadea,5); //partiendo de 5 hasta el final, por no especificar el último


$cadea="evanaveira.clases@gmail.com";
echo "<br> Obter desde un caracter hasta el final:". strstr($cadea,"@");

$cadea="En un lugar de La Mancha de cuyo nombre no quiero acordarme.";
echo "<br> Contar palabras:".str_word_count($cadea);  //12 palabras
echo "<br>Comparar dos string: ".strcmp("hola", "Hola");
echo "<br>Comparar dos string: ".strcmp("hola", "hola");
echo "<br>Comparar dos string: ".strcmp("hola", "adios");

?>
```

Exercicios.

3.2.2. Arrays

Un **array** é un **tipo especial de variable** que pode conter máis dun valor ao mesmo tempo. **Defínense mediante o uso da función array()**. En PHP existen **tres tipos de arrays**:

- **Arrays indexados: os elementos indéxanse cun número**. Exemplo:

Exemplo de arrays indexados

```
<?php
$cars=array("Volvo","BMW","Toyota");
echo "Gústanme os coches " . $cars[0] . ", " . $cars[1] . " e " . $cars[2] . ".";
?>
```

- **Arrays asociativos: os elementos indéxanse con claves**. Exemplo:

Exemplo de arrays asociativos

```
<?php
$age=array("Uxío"=>"35","Xoán"=>"37","Miguel"=>"43");
echo "Uxío ten " . $age['Peter'] . " anos.";
?>
```

- **Arrays multidimiensionais: son arrays que conteñen á súa vez outros arrays**. Exemplo:

Exemplo de arrays multidimensionais

```
<?php
// Array de dúas dimensións:
$familias = array
  (
  "Vázquez"=>array
```

```
(
"Carmen",
"Pilar",
"Juan"
),
"García"=>array
(
"Manuel"
),
"López"=>array
(
"Nuria",
"Patricia",
"Alberto"
)
);

echo "Es " . $familias['Vázquez'][2] .
" parte de la familia Vázquez?";

// Resultado: Es Juan parte da familia Vázquez?
?>
```

Para obter a **lonxitude dun array**, é dicir, o número de elementos dun array, úsase a **función count()**. A continuación amósase un exemplo de uso desta función.

Exemplo de uso da función count()

```
<?php
$coches=array("Volvo","BMW","Toyota");
echo count($coches);

//Resultado: 3
?>
```

Creación/modificación con sintaxe de corchetes cadrados

É posible modificar unha matriz existente ao definir valores explicitamente nela.

Isto é posible ao asignar valores á matriz ao mesmo tempo que se especifica a clave entre corchetes. Tamén é posible omitir a clave, agregar unha parella baleira de corchetes ("[]") ao nome da variable nese caso.

```
$matriz[clave] = valor;
$matriz[] = valor;
// clave pode ser un integer ou string
// valor pode ser calquera valor
```

Se *$matriz* non existe aínda, será creada. De modo que esta é tamén unha forma alternativa de especificar unha matriz. Para modificar un certo valor, simplemente asignase un novo valor a un elemento especificado coa súa clave. Para eliminar unha parella clave/valor, necesitase eliminala mediante *unset()*.

```
<?php
$matriz = array(5 => 1, 12 => 2);
$matriz[] = 56;    // Isto é igual que $matriz[13] = 56;
                   // neste punto do script
$matriz["x"] = 42; // Isto agrega un novo elemento á
                   // matriz coa clave "x"
unset($matriz[5]); // Isto elimina o elemento da matriz
unset($matriz);    // Isto elimina a matriz completa
```

```
?>
```

Como se menciona anteriormente, se prové os corchetes sen ningunha clave especificada, entón tómase o máximo dos índices enteiros existentes, e a nova clave será ese valor máximo + 1. Se non existen índices enteiros aínda, a clave será 0 (cero). Se especifica unha clave que xa tiña un valor asignado, o valor será substituído.

A clave enteira máxima usada para este caso non necesita existir actualmente na matriz. Tan só debeu existir na matriz nalgún momento desde que a matriz fose re-indexada. O seguinte exemplo ilustra este caso:

```
<?php
// Crear unha matriz simple.
$matriz = array(1, 2, 3, 4, 5);
print_r($matriz);
// Agora eliminar cada ítem, pero deixar a matriz mesma intac-ta:
foreach ($matriz as $i => $valor) {
    unset($matriz[$i]);
}
print_r($matriz);
// Agregar un ítem (note que a nova clave é 5, en lugar de 0 como
// podería esperarse).
$matriz[] = 6;
print_r($matriz);
// Re-indexar:
$matriz = array_values($matriz);
$matriz[] = 7;
print_r($matriz);
?>
O resultado do exemplo seria:
Array
(
    [0] => 1
    [1] => 2
    [2] => 3
    [3] => 4
    [4] => 5
)
Array
(
)
Array
(
    [5] => 6
)
Array
(
    [0] => 6
    [1] => 7
```

```
)
```

Funcións útiles

Existe un bo número de funcións útiles para traballar con matrices.

A función *unset()* permite eliminar a definición de claves dunha matriz. Hai que ter en conta que a matriz NON é re-indexada. Se só usa "índices enteiros comúns" (comezando desde cero, incrementando de un en un), pódese conseguir o efecto de re-indexación usando *array_values()*.

```
<?php
$a = array(1 => 'un', 2 => 'dous', 3 => 'tres');
unset($a[2]);
/* producirá unha matriz que fose definida como
   $a = array(1 => 'un', 3 => 'tres');
   e NON
   $a = array(1 => 'un', 2 =>'tres');
*/
$b = array_values($a);
// Agora $b é array(0 => 'un', 1 =>'tres')
?>
```

A estrutura de control *foreach* existe especificamente para as matrices. Esta prové unha maneira fácil de percorrer unha matriz.

array explode (string $separador, string $cadea [, int $limite])

Devolve unha matriz de cadeas, cada unha das cales é unha subcadea de cadea formada mediante a súa división nas fronteiras marcadas pola cadea separador. Se se especifica limite, a matriz devolta conterá un máximo elementos co último contendo o resto da cadea.

Se separador é unha cadea baleira (""), *explode()* devolve un valor igual a FALSE. Se separador contén un valor que non está presente en cadea, a función *explode()* devolve unha matriz que contén a cadea.

```
<?php
// Ejemplo 1
$pizza  = "trozo1 trozo2 trozo3 trozo4 trozo5 trozo6";
$trozos = explode(" ", $pizza);
echo $trozos[0]; // trozo1
echo $trozos[1]; // trozo2
?>
```

string implode (string $elemento_union, array $elementos)

Devolve unha cadea que contén unha representación de todos os elementos da matriz no mesmo orde, pero coa cadea elemento_union no medio dos mesmos.

```
<?php
$array = array('apellido', 'email', 'telefono');
$separado_por_comas = implode(",", $array);
```

```
echo $separado_por_comas; // apellido,email,telefono
?>
```

int count (mixed $var [, int $mode])

Conta os elementos dunha matriz ou propiedades dun obxecto

Devolve o número de elementos en var, que tipicamente é un array, porque calquera outra cousa diferente dun obxecto tería só un elemento.

Se o parámetro opcional *mode* é iniciado a COUNT_RECURSIVE (ou 1), *count()* contará recursivamente a matriz. Isto é útil particularmente para contar todos os elementos dunha matriz multidimensional. O valor por defecto para *mode* é 0.

void sort (array &$matriz [, int $sort_flags])

Esta función ordena unha matriz. Os elementos estarán ordenados de menor a maior cando a función termine.

Asigna novos índices na matriz.

O parámetro opcional *sort_flags* pode ser usado para modificar o comportamento do ordenamento usando estes valores:

```
SORT_REGULAR - comparación normal (non cambia os tipos)
SORT_NUMERIC - comparación numérica
SORT_STRING - comparación por cadeas
SORT_LOCALE_STRING - compara elementos como cadeas, ba-seado na localización
actual
```

void rsort (array &$matriz [, int $sort_flags])

Esta función ordena unha matriz en orden inverso, de maior a menor.

Recomendacións sobre matrices e cousas a evitar

Porqué é incorrecto $foo[bar]?

Sempre deben usarse comiñas ao redor dun índice de matriz tipo cadea literal. Por exemplo, usarase *$foo['bar']* e non *$foo[bar]*. Pero que está mal en *$foo[bar]*? É posible ver a seguinte sintaxe en scripts vellos:

```
<?php
$foo[bar] = 'inimigo';
echo $foo[bar];
// etc
?>
```

Isto está mal, pero funciona. Entón, porqué está mal? A razón é que este código ten unha constante indefinida (*bar*) en lugar dunha cadea (*'bar'* - coas comiñas), e poida que no futuro PHP defina constantes que, desafortunadamente para este código, teñan o mesmo nome. Funciona porque PHP automaticamente converte unha cadea pura (unha cadea sen comiñas que non corresponda con símbolo coñecido algún) nunha cadea que contén a cadea pura. Por exemplo, se non se definiu unha constante chamada bar, entón PHP substituirá o seu valor pola cadea *'bar'* e usará esta última.

Isto non quere dicir que sempre haxa que usar comiñas na clave. Non se necesitará usar comiñas con claves que sexan constantes ou variables, xa que en tal caso PHP non poderá interpretar os seus valores.

```
<?php
error_reporting(E_ALL);
ini_set('display_errors', true);
ini_set('html_errors', false);
// Matriz simple:
$matriz = array(1, 2);
$calculo = count($matriz);
for ($i = 0; $i < $calculo; $i++) {
    echo "\nRevisando $i: \n";
    echo "Mal: " . $matriz['$i'] . "\n";
    echo "Ben: " . $matriz[$i] . "\n";
}
?>
```

O resultado do exemplo seria:

```
Revisando 0:
Notice: Undefined index:  $i in /path/to/script.html en liña 9
Mal:
Ben: 1
Revisando 1:
Notice: Undefined index:  $i in /path/to/script.html en liña 9
Mal:
Ben: 2
```

Máis exemplos para demostrar este feito:

```
<?php
// Mostrar todos os erros
erro_reporting(E_ALL);
$matriz = array('froita' => 'mazá', 'vexetal' => 'cenoria');
// Correcto
print $matriz['froita'];   // mazá
print $matriz['vexetal']; // cenoria
// Incorrecto. Isto funciona pero tamén xera un erro de PHP de //nivel
E_NOTICE xa que non hai definida unha constante chamada froita
//
// Notice: Use of undefined constant froita - assumed 'froita' in...
print $matriz[froita];    // mazá
// Definamos unha constante para demostrar o que pasa. Asigna-remos o
// valor 'vexetal' a unha constante chamada froita.
define('froita', 'vexetal');
// Note a diferenza agora
print $matriz['froita'];  // mazá
print $matriz[froita];    // cenoria
```

```
// O seguinte está ben xa que se atopa ao interior dunha
// cadea. As constantes non son procesadas ao interior de //
cadeas, así que non se produce un erro E_NOTICE aquí
print "Ola $matriz[froita]";  // Ola mazá
// Cunha excepción, os corchetes que rodean as matrices ao
// interior de cadeas permiten o uso de constantes
print "Ola {$matriz[froita]}";    // Ola cenoria
print "Ola {$matriz['froita']}";  // Ola mazá
// Isto non funciona, resulta nun erro de intérprete como:
// Parse erro: parse erro, expecting T_STRING' or T_VARIABLE' or
T_NUM_STRING'
// Isto aplícase tamén ao uso de superglobales en cadeas, por suposto
print "Ola $matriz['froita']";
print "Ola $_GET['foo']";
// A concatenación é outra opción
print "Ola " . $matriz['froita']; // Ola mazá
?>
```

Cando habilita *error_reporting()* para mostrar erros de nivel E_NOTICE (por exemplo definindo o valor E_ALL) verá estes erros. Por defecto, error_reporting atópase configurado para non mostralos.

Tal e como se indica na sección de sintaxe, debe existir unha expresión entre os corchetes cadrados ('[' e ']'). Iso quere dicir que pode escribir cousas como esta:

```
<?php
echo $matriz[algunha_funcion($bar)];
?>
```

Este é un exemplo do uso dun valor devolto por unha función como índice de matriz. PHP tamén coñece as constantes, tal e como puido apreciar aquelas E_* antes.

```
<?php
$descricións_de_erro[E_ERROR]   = "Un erro fatal ha ocorrido";
$descricións_de_erro[E_WARNING] = "PHP produciu unha advertencia";
$descricións_de_erro[E_NOTICE]  = "Esta é unha noticia informal";
?>
```

Note que E_ERROR é tamén un identificador válido, así como bar no primeiro exemplo. Pero o último exemplo é equivalente a escribir:

```
<?php
$descricións_de_erro[1] = "Un erro fatal ha ocorrido";
$descricións_de_erro[2] = "PHP produciu unha advertencia";
$descricións_de_erro[8] = "Esta é unha noticia informal";
?>
```

xa que E_ERROR é igual a 1, etc.

Tal e como o explicamos nos anteriores exemplos, $foo[bar] funciona pero está mal. Funciona, porque debido á súa sintaxe, espérase que bar sexa unha expresión constante. Con

todo, neste caso non existe unha constante co nome bar. PHP asume agora que se quixo dicir bar literalmente, como a cadea "bar", pero que esqueceu escribir as comiñas.

Entón porqué está mal?

Nalgún momento no futuro, o equipo de PHP pode querer usar outra constante ou palabra clave, ou poida que se introduza outra constante na súa aplicación, e entón aparecerán problemas. Por exemplo, neste momento non pode usar as palabras *empty* e *default* desta forma, xa que son palabras clave reservadas especiais.

Exemplos

O tipo matriz en PHP é bastante versátil, así que aquí preséntanse algúns exemplos que demostran o poder completo das matrices.

```
<?php
$a = array( 'cor'  => 'vermella',
            'sabor'  => 'doce',
            'forma'  => 'redonda',
            'nomee' => 'mazá',
                       4        // a clave será 0
          );
// é completamente equivalente a
$a['cor']  = 'vermello';
$a['sabor']  = 'doce';
$a['forma']  = 'redonda';
$a['nome'] = 'mazá';
$a[]         = 4;        // a clave será 0
$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// resultará na matriz array(0 => 'a' , 1 => 'b' , 2 => 'c'),
// ou simplemente array('a', 'b', 'c')
?>


<?php
// Array como mapa de propiedades
$mapa = array( 'versión'          => 4,
               'SO'               => 'Linux',
               'idioma            => 'inglés',
               'etiquetas_curtas' => true
            );
// claves estritamente numéricas
$matriz = array( 7,
                 8,
                 0,
                 156,
                 -10
               );
// isto é o mesmo que array(0 => 7, 1 => 8, ...)
```

```
$cambios = array( 10, // clave = 0
                  5    =>  6,
                  3    =>  7,
                  'a'  =>  4,
                  11, // clave = 6 (o índice enteiro máximo era 5)
                  '8'  =>  2, // clave = 8 (enteiro!)
                  '02' => 77, // clave = '02'
                  0  => 12  // o valor 10 será substituído por 12
                );

// matriz baleira
$vacio = array();
?>
<?php
$cores = array('vermello', 'azul', 'verde', 'amarelo');
foreach ($cores as $cor) {
    echo "Gústalle a $cor?\n";
}
?>
O resultado do exemplo seria:
Gústalle o vermello?
Gústalle o azul?
Gústalle o verde?
Gústalle o amarelo?
```

Modificar os valores da matriz directamente é posible a partir de PHP 5, pasándoos por referencia. As versións anteriores necesitan unha solución alternativa:

```
<?php
// PHP 5
foreach ($cores as &$cor) {
    $cor = strtoupper($cor);
}
unset($cor); /* asegúrase de que escrituras subseguintes a $cor
non modifiquen o último elemento da matriz */
// Alternativa para versións anteriores
foreach ($cores as $clave => $cor) {
    $cores[$clave] = strtoupper($cor);
}
print_r($cores);
?>
```

O resultado do exemplo seria:

```
Array
(
    [0] => VERMELLO
    [1] => AZUL
```

```
    [2] => VERDE
    [3] => AMARELO
)
```

Este exemplo crea unha matriz con base un.

```
<?php
$primercuarto  = array(1 => 'Xaneiro', 'Febreiro', 'Marzo');
print_r($primercuarto);
?>
```

O resultado do exemplo seria:

```
Array
(
    [1] => 'Xaneiro'
    [2] => 'Febreiro'
    [3] => 'Marzo'
)
<?php
// encher unha matriz con todos os ítems dun directorio
$xestor = opendir('.');
while (false !== ($arquivo = readdir($xestor))) {
    $arquivos[] = $arquivo;
}
closedir($xestor);
?>
```

Dado que o valor dunha matriz pode ser calquera cousa, tamén pode ser outra matriz. Desta forma é posible crear matrices recursivas e multi-dimensionales.

```
<?php
$froitas = array ( "froitas"  => array ( "a" => "laranxa",
                                         "b" => "banana",
                                         "c" => "mazá"
                                       ),
                   "numeros" => array ( 1,
                                        2,
                                        3,
                                        4,
                                        5,
                                        6
                                       ),
                   "buratos"   => array (    "primeiro",
                                         5 => "segundo",
                                              "terceiro"
                                       )
                 );
```

```
// Algúns exemplos que fan referencia aos valores da matriz anterior
echo $froitas["buratos"][5];    // imprime "segundo"
echo $froitas["froitas"]["a"];  // imprime "laranxa"
unset($froitas["buratos"][0]);  // elimina "primeiro"
// Crear unha nova matriz multi-dimensional
$froitas["mazá"]["verde"] = "ben";
?>
```

Debe advertirse que a asignación de matrices sempre involucra a copia de valores. Necesitase usar o operador de referencia para copiar unha matriz por referencia.

```
<?php
$matriz1 = array(2, 3);
$matriz2 = $matriz1;
$matriz2[] = 4; // $matriz2 cambia,
                // $matriz1 segue sendo array(2, 3)
$matriz3 = &$matriz1;
$matriz3[] = 4; // agora $matriz1 e $matriz3 son iguais
?>
```

Exercicios.