PHP

Índice

Estructuras básicas

IF

```
if( condición ){
    // Lo que pasa si vale la condición
} else {
    // Lo que pasa si no vale la condición
}
```

FOR

```
for( asignación; condición; modificación ){
    // Cuerpo del ciclo
}
```

Asignación

Es la instrucción que se ejecuta una sola vez antes del ciclo, generalmente se usa para asignar un valor inicial a la variable índice.

Condición

Es la condición para que se ejecute el cuerpo del ciclo, si la condición no se cumple antes de realizar una repetición entonces termina el ciclo.

Modificación

Es la instrucción que se encarga de modificar la variable índice y se ejecuta cada vez que termina una repetición del ciclo.

Ejemplo:

```
for($i = 0; $i < 15; $i++){
    echo( $i );
}
```

En este ejemplo, la asignación es `$i = 0` creando una variable `$i` y dándole el valor `0`. La condición es `$i < 15`, indicando que cuando `$i` ya no sea menor a `15` terminará el ciclo. Por último, la modificación es `$i++`, indicando que luego de cada repetición, `$i` se incrementará en `1`. Finalmente, este for recorrerá todos los números del `1` al `14` y los mostrará.

Ejemplo 2:

```
$suma = 0;
for($i = 10; $i ⩾ 0; $i -= 2){
    $suma += $i;
}
```

En este segundo ejemplo, la asignación es `$indice = 10`, creando una variable `$indice` y dándole el valor `10`. La condición es `$indice ⩾ 0`, indicando que cuando `$indice` valga `0` o menos terminará el ciclo. Por último, la modificación es `$i -= 2`, indicando que luego de cada repetición, `$indice` se decrementará en `2`. Finalmente este código recorrerá todos los números pares del `10` al `0` y los sumará en la variable `$suma`.

FOREACH

```
foreach( asignación ){
    // Cuerpo del ciclo
}
```

Asignación foreach

Existen 2 tipos de asignación en un ciclo foreach: en la primera se asigna una única variable la cual contendrá el valor de cada uno de los elementos; en la segunda se asignan 2 valores, la clave (valor con el que se accede en el arra) y el valor (lo que está almacenado en la posición valor del array). **Importante:** modificar la variable que contiene el valor no modificará el array.

Ejemplo:

```
$miarray = array(10, 5, 8, 7, 6);
foreach($mirray as $valor){
    echo( $valor );
}
```

En este ejemplo al escribir `$miarray as $valor`, indicamos que la variable `$valor` contendrá el valor de un elemento del array en cada repetición. Finalmente, este `foreach` mostrará todos los valores de `$miarray` en el orden en que se encuentran en el mismo.

Ejemplo 2:

```
$miarray = array("hola", "chau", "foreach", "test", "prueba");
foreach($miarray as $clave ⇒ $valor){
    echo( "En el lugar $clave del array se encuentra el valor $valor <br>" );
}
```

En este ejemplo al escribir `$miarray as $clave ⇒ $valor`, indicamos que la variable `$clave` tendrá el valor con el que se accede al array y `$valor` contendrá el valor de un elemento del array en cada repetición. Finalmente, este `foreach` tendrá el siguiente output:

```
En el lugar 0 del array se encuentra el valor hola
En el lugar 1 del array se encuentra el valor chau
En el lugar 2 del array se encuentra el valor foreach
```

```
En el lugar 3 del array se encuentra el valor test
En el lugar 4 del array se encuentra el valor prueba
```

WHILE

```
while( condición ){
    // Cuerpo del ciclo
}
```

Tipos de datos implícitos

Int Un número entero que puede ser negativo o positivo.

Float Un número con coma, también puede ser negativo o positivo.

Bool Un valor de verdad, puede ser true o false. Se pueden utilizar directamente como condiciones en todas las estructuras que las utilizan.

String (funciones y operadores para strings) Una cadena de caracteres. Siempre se crean poniendo su contenido entre comillas. Podemos pensar a estos como un array especial que sólo contiene caracteres.

Operadores y funciones sobre strings **'.'**: este operador nos permite concatenar dos strings, por ejemplo:

```
$a = "hola";
$b = "chau";
$c = $a . $b;

echo( $c ); // "holachau"
```

'•=': este operador utiliza el operador '.' uniendo el string de la derecha al ya existente de la izquierda, por ejemplo:

```
$a = "hola";
$b = "chau";
$a •= $b;

echo( $a ); // "holachau"
```

'[]': este operador nos permite acceder a un caracter específico de un string, por ejemplo:

```
$a = "abcdef";

echo( $a[3] ); // "d"
```

'strlen(string)': esta función devuelve el largo de un string, por ejemplo:

```
$a = "hola";

echo( strlen($a) ); // 4
```

'substr(string, inicio, cantidad)': esta función devuelve un substring (parte de un string) empezando en la posición inicio y del largo cantidad, por ejemplo:

```
$a = "abcdef";

echo( substr($a, 2, 2) ); // "cd"
```

También se puede omitir el largo, entonces se calculará el substring desde la posición inicio hasta el final del string original:

```
echo( substr($a, 2) ); // "cdef"
```

También se pueden utilizar valores de inicio negativos, entonces se contará la posición inicio desde el final del string:

```php
echo( substr($a, -2, 2) ); // "ef"
```

Más funciones ...

ARRAY (funciones para strings)

Una lista de valores, la cual se puede recorrer para acceder a cada uno de ellos individualmente.

Operadores y funciones sobre arrays

'array(valores)': esta función nos permite crear un array con los valores especificados, por ejemplo:

```php
$a = array(1, 5, 6, 10);

echo( $a ); // [1, 5, 6, 10]
```

'[]': este operador nos permite acceder a un valor específico de un array, por ejemplo:

```php
$a = array("hola", "chau", "test", "prueba");

echo( $a[2] ); // test
```

'count(array)': esta función permite ver la cantidad de elementos que contiene el array pasado como parámetro, por ejemplo:

```php
$a = array(1.6, 1.9, 5.8);

echo( count($a) ); // 3
```

'array_push(array, valor)': esta función permite agregar un elemento al final del array, por ejemplo:

```php
$a = array(5, 9, 40, 7);
array_push($a, 100);

echo( $a ); // [5, 9, 40, 7, 100]
```

Arrays asociativos

Los arrays en PHP tienen una particularidad, si bien podemos utilizarlos como arrays normales de cualquier otro lenguaje de programación, también podemos utilizarlos como los llamados "arrays asociativos". Estos arrays pueden tener como claves otro tipo de dato que no sea un entero.

Ejemplo de creación de un array asociativo:

```php
$a = array("Nacho" => 22, "Juan" => 30, "Miguel" => 52);
```

Ejemplo de acceso a un elemento:

```php
echo( $a["Juan"] ); // 30
```

Funciones

```php
function nombreDeFuncion( parámetros ){
    // Cuerpo de la función
}
```

Podemos pensar en las funciones como máquinas que realizan una operación sobre los parámetros que les damos. Nos ayudan a hacer el código más legible y a reutilizar código sin tener que volver a escribirlo. Por ejemplo, si tuviera un arreglo al cual lo quiero ordenar y luego mostrar solo los valores ubicados en índices pares podría hacer lo siguiente:

```php
$a = array(5, 7, 3, 9);

// Código para ordenar (unas 15 líneas)

for($i = 0; $i < count($a); $i += 2){
    echo( $a[$i] );
}
```

Luego, si quisiera mostrar realizar el mismo procedimiento pero en un array `$b`, debería escribir nuevamente todo el código. Además, cada vez que leo el código anterior, debo tomarme un tiempo para entender lo que hace. Si utilizáramos funciones podríamos hacer lo siguiente:

```php
function ordenar($arrayAOrdenar){
// Código de ordenar el array
return $arrayAOrdenar;
}
    function mostrarPosPares($arrayAMostrar){
    for($i = 0; $i < count($a); $i += 2){
        echo( $a[$i] );
    }
}
```

Entonces, para cada array al cual queremos realizarle este procedimiento deberíamos escribir únicamente lo siguiente:

```php
$a = array(5, 7, 3, 9);
$a = ordenar($a);
mostrarPosPares($a);
```

Además, se puede notar cómo se entiende automáticamente qué hace este código, ya que los nombres de las funciones nos permiten abstraernos del procedimiento interno y utilizarlo como una caja negra.

Parámetros

Los parámetros son los valores que recibe una función, y con los cuales operará. Una vez que se llama a una función con determinados valores, ésta creará las variables de sus parámetros de acuerdo al orden en que se presentan los valores, una vez que termina la función estas variables desaparecen, y no pueden ser accedidas fuera de la misma. A este concepto se le llama scope o alcance de las variables, y explica en qué rango del código existe una variable dada. Por ejemplo:

```php
function sumaTotal($array){
    $total = 0;
    foreach($array as $value){
        $total += $value;
    }
    return $total;
}
// En esta línea $total no existe porque está fuera de la función
echo( $total );

// En esta línea $array no existe porque está fuera de la función
echo( $array );
```

Este concepto de scope también se aplica a las estructuras como ciclos, es decir, cualquier variable que se cree dentro de un ciclo for (por ejemplo), no se podrá acceder por fuera ya que se destruirá una vez terminado el ciclo.

Return y tipos de retorno

Todas las funciones de PHP pueden separarse en 2 tipos: funciones con retorno y funciones sin retorno. En programación, a las funciones sin retorno se las llama `void`, mientras que a las funciones con retorno se las define por el tipo de dato que devuelven. Dentro de una función, una vez que se utiliza la sentencia return, se termina automáticamente la ejecución de la misma. Por ejemplo:

```php
function test($a, $b){
    return $a + $b;
    echo( "Esto no se ejecuta");
}
```

Esta característica se puede aprovechar a nuestro favor a la hora de crear funciones. Si tuviésemos nuestro return en una estructura condicional, sabemos que lo demás se ejecutará solo si no valía la condición anterior, por ejemplo:

```php
function estaEnArray($array, $buscado) {
    foreach ($array as $elemento) {
        if ($elemento == $buscado) {
            return true;
        }
    }
    return false;
}
```

En este ejemplo sabemos que si termina el `foreach`, es que nunca realizamos un `return true`, por lo tanto, el elemento no se encuentra en el array.

También podemos aprovechar esto para realizar validaciones sobre los datos:

```php
function sumaTotal($array){
    if(!is_array($array)){
        return -1;
    }

    $total = 0;
    foreach($array as $valor){
        $total += $valor;
    }
    return $total;
}
```

En este caso, queremos retornar `-1` si el parámetro que se envía no es un array, para evitar que el código tire error. (Notar que esta forma de avisar no es válida si el array puede tener números negativos)

Cosas útiles

Función echo

La función echo nos permite escribir strings en el lugar donde la llamamos. Es importante notar que cuando uno llama a la función con parámetros que no son strings, éstos se convierten en strings de una manera determinada dependiendo de su tipo de datos. Importante: las variables booleanas no cuentan con esta forma rápida de transformarse en string, por lo que no serán mostradas en un `echo`.

Función var_dump

Una función parecida a `echo`, pero que recibe cualquier tipo de variable y muestra información importante de la misma. Se utiliza mucho para mostrar arrays asociativos.

Función include / include_once

Esta función nos permite incluir otro archivo de PHP en el archivo actual, brindándonos la posibilidad de tener una mejor distribución de nuestras funciones y constantes globales y permitiendo, además, reutilizar código.

```php
include_once( ruta del archivo );
```

Rutas absolutas

Las rutas absolutas son una forma de ubicar un archivo en una computadora/servidor de manera global, es decir, desde cualquier ubicación arbitraria, es posible acceder al archivo especificado por medio de estas. Las

utilizamos todos los días por ejemplo en el explorador de archivos, por ejemplo, todos tenemos los archivos para correr PHP en 'C:\AppServ\www' la cual es una ruta absoluta.

Rutas relativas

Las rutas relativas, en cambio, son una forma de ubicar un archivo desde una ubicación específica. Podemos pensarlo como las direcciones que le damos a alguien que ya está ubicado en una ruta, para llegar a la que queremos. Para esta tarea, utilizaremos los siguientes comandos:

'./': Este comando representa la carpeta actual del archivo en cuestión. **'../'**: Este comando representa la carpeta padre de la carpeta actual.

Ejemplo: Si el árbol de carpetas fuera el siguiente:

Si nosotros estamos en el archivo `index.php`, entonces para acceder a la carpeta test debemos utilizar la ruta relativa `../test/`, la cual significa subir una carpeta en el árbol y luego entrar a la carpeta `test`.

Ejemplo de include con ruta relativa:

Si estamos ubicados en el archivo `index.php` y queremos incluir `functions.php`:

```
include_once("../test/functions.php");
```

Extras (100% opcionales)

Operadores ternarios

Los operadores son una forma de escribir "una suerte de if" (notar la letra y las comillas) de manera corta, sirven para evaluar una condición y devolver un valor dependiendo de si el resultado es verdadero o falso. Se escriben de la siguiente manera:

```
echo( condition ? "resultado si es verdadero" : "resultado si es falso" );
```

Ejemplo:

```
$a = 5;
echo( $a < 10 ? "a es menor a 5" : "a es mayor 5" );
```