



# RECORRIDO DE ARREGLOS



¿Cómo recorreremos los elementos de un arreglo?  
Por ejemplo para obtener el mayor elemento o  
el elemento que cumpla una condición



# Recorrido de Arreglos

Para recorrer un arreglo se utilizan las **estructuras de control Repetitivas**

(**Recorrido Exhaustivo**)



**Acceder a TODOS los elementos del arreglo** desde el elemento 0 al elemento N-1, siendo N la cantidad de elementos de un arreglo.

La estructura de control más adecuada es la instrucción **FOR / FOREACH**.

(**Recorrido Parcial**)



**Recorrer el arreglo hasta encontrar un elemento** con determinadas características, la estructura más adecuada es **WHILE**, siendo la condición de corte la obtención del elemento buscado (**y si dicho elemento no existe, controlamos no acceder a una posición que no existe!**)



# Recorrido de Arreglos

**Recorrido Exhaustivo:  
Estructura de control FOR**

```
$arreglo=[31,21,12,1,5];  
$n = count($arreglo);  
  
for($i=0; $i<$n; $i++){  
    ...  
    instruccion sobre $arreglo[$i];  
    ...  
}
```

**Ejercicio:** Obtener el promedio de elementos del arreglo



# Recorrido de Arreglos

## Recorrido Exhaustivo: Estructura de control FOREACH

```
$arreglo=[31,21,12,1,5];
```

```
foreach($arreglo as $indice => $elemento){  
    echo "el elemento de la posicion".  
        $indice." es ".$elemento."\n";  
}
```

Sintaxis 1

```
foreach($arreglo as $elemento){  
    echo "elemento es ".$elemento."\n";  
}
```

Sintaxis 2



# Recorrido de Arreglos

## Recorrido Exhaustivo: Estructura de control FOREACH

```
$unEmpleado = [ "nombre" => "Juan" ,  
                 "apellido" => "Perez",  
                 "antiguedad" => 3 ,  
                 "fechaNac" => "12/06/79"] ;
```

Inicialización

```
foreach($unEmpleado as $indice => $dato){  
    echo $indice.": ".$dato. "\n";  
}
```

Recorrido  
Exhaustivo



# Recorrido de Arreglos

Recorrido Parcial:

Estructura de control MIENTRAS

```
$n = count($arreglo);  
$i = 0;  
  
while($i < $n && condición) {  
    ....  
    instrucción sobre el arreglo[$i];  
    ....  
    $i = $i + 1;  
}
```

## Ejercicio:

Escribir la posición del primer número par contenido en un arreglo

Ejemplos:

1	3	8	5
0	1	2	3

1	11	9	5	9
0	1	2	3	4



# Ejercicios:

- Recorrido exhaustivo: **Sumar todos los elementos de un arreglo que contiene valores enteros.**
- Recorrido parcial: **Sumar los números de un arreglo hasta encontrar el primer número negativo.**

**Ejemplo:**

arreglo =

10	20	-20	1
0	1	2	3

} valores  
} índices

count(arreglo) = 4



# Funciones de Ordenamiento PHP:

Los Arrays se pueden ordenar utilizando las siguientes funciones

**sort:** Ordena los elementos de menor a mayor. Elimina cualquier clave existente y asigna nuevos índices a partir del 0.

**<?php**

```
$frutas=["l"=>"limon", "n"=>"naranja", "m"=>"manzana", "d"=>"durazno"];
```

```
sort($frutas);
```

```
foreach($frutas as $indice => $unafruta){  
    echo"$indice=$unafruta\n";  
}
```

**?>**

**rsort:** Similar sort, pero ordena los elementos de mayor a menor.





# Funciones de Ordenamiento PHP:

**asort:** Ordena los elementos de menor a mayor. Mantiene la correlación de los índices con los elementos con los que está asociado.

```
<?php
```

```
$frutas=["l"=>"limon", "n"=>"naranja", "m"=>"manzana", "d"=>"durazno"];
```

```
asort($frutas);
```

```
foreach($frutas as $indice => $unafruta){  
    echo"$indice=$unafruta\n";}
```

```
}
```

```
?>
```

**arsort:** Similar asort, Pero ordena los elementos de mayor a menor.



# Funciones de Ordenamiento PHP:

**ksort:** Ordena por clave de menor a mayor. Mantiene la correlación de la clave con el elemento con los que está asociado.

```
<?php
```

```
$frutas=["o"=>"limon", "n"=>"naranja", "m"=>"manzana", "d"=>"durazno"];
```

```
ksort($frutas);
```

```
foreach($frutas as $indice => $unafruta){  
    echo"$indice=$unafruta\n";}
```

```
}
```

```
?>
```

**krsort:** Similar ksort, Pero ordena las claves de mayor a menor.



# Funciones de Ordenamiento PHP:

## Ordenamiento definido por el usuario

**uasort:** Ordena los elementos usando una función de comparación definida por el usuario. Mantiene la correlación de los índices con los elementos con los que está asociado.

**<?php**

// Función de comparación

**function cmp(\$a, \$b) {**

if (\$a == \$b){

    \$orden = 0;}

elseif(\$a < \$b){

    \$orden = -1;}

else{

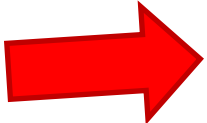
    \$orden = 1;}

**return \$orden;**

**}**

Función definida para  
comparar los elementos  
del arreglo.

Debe retornar 0, -1 o 1

 \$arreglo=['a' => 4, 'b' => 8, 'c' => -1, 'd' => -9 ];

**uasort(\$arreglo, 'cmp');**

**foreach**(\$arreglo as \$indice=>\$elemento){

    echo "**\$indice = \$elemento \n**";

**}**



# Funciones de Ordenamiento PHP:

<http://php.net/manual/es/array.sorting.php>



# String: Cadena de caracteres



# String

## ■ Asignar un String

- `$a = "una cadena \n."`
- `$b = 'otra cadena \n.'`

## ■ Diferencia entre ' ' y " " (comillas simples y dobles)

- `echo "la variable a contiene el valor: $a \n"`
- `echo ' la variable a contiene el valor: $a \n'`
- `echo "la variable b contiene el valor: $b "`
- `echo 'la variable b contiene el valor:' . $b`

*Las ' ' no permite que se expandan los caracteres especiales  
ni los valores de valores de variables*

## ■ Comparación de string

- `"aaa" < "aab"` (orden alfabético)
- `"aab" >= "aaa"`
- `"aaa" == "aaa"`

`"a" < "b"`  
`"b" < "c"`  
`"c" < "d"`  
Etc...



# String: cadena de caracteres

## ■ Funciones PHP para string

- `trim(" hola que tal ")` //quita los espacios en blanco de izq y der.
- `strtoupper("aaaaBbbb")` //convierte a mayúsculas
- `strtolower("AAAABBBB")` //convierte a minúsculas
- `strlen("varios caracteres")` //cantidad de caracteres del string.
- `explode(" ", "me encanta programar")` //arreglo de componentes separados por un delimitador.
- `substr("mi casa", 4,3)` //devuelve una parte de la cadena

## ■ Más funciones en:

<http://php.net/manual/es/book.strings.php>



# String: Cadena de caracteres

`$cadena = "programar";`

Internamente las cadenas de caracteres se almacenan de manera similar a un arreglo indexado, con la salvedad que para string **NO se utiliza la función count**, sino que se **utiliza la función strlen** para determinar la cantidad de caracteres:

Representación:

`$cadena =`

p	r	o	g	r	a	m	a	r
0	1	2	3	4	5	6	7	8

```
$cadena = "programar";           //instrucción de asignación
$longitud = strlen($cadena);     //cantidad de caracteres del string.
echo $longitud ;                 //imprime 9.
```





# String: cadena de caracteres

## Podemos recorrer un string como un arreglo.

### ■ Recorrer cada letra del String

```
$cadena = "Me encanta programar!";  
  
for ($i=0; $i<strlen($cadena); $i++){  
    echo "letra $i: " . $cadena[$i] . "\n";  
}
```



## String: Ejercicio

Implementar una función cuyos parámetros de entrada es una Cadena de Caracteres y una Letra. Debe retorna la primera posición de la Cadena de Caracteres en la que se encuentra La letra. Si la letra no existe retornar -1

Pregunta: ¿Utilizará un recorrido exhaustivo o un recorrido parcial?