

DESARROLLO WEB EN ENTORNO SERVIDOR

CAPÍTULO 5:

**Programación basada en lenguajes de marcas con
código embebido en el servidor.**

Arrays o Matrices

Arrays o Matrices

- Los arrays o matrices son estructuras que permiten el almacenamiento de un conjunto de datos.
- Son variables que admiten varios valores.
- Definición: un array o matriz es un conjunto ordenado de elementos identificados por un índice (la posición del elemento dentro de esta colección ordenada), de modo que en cada posición marcada por un índice el array contiene un valor.

```
<?php
```

```
// define una matriz
```

```
$frutas = array('manzana', 'plátano', 'piña', 'uva');
```

```
?>
```

- Para recuperar un valor determinado se recurre a su índice tal que: \$frutas [0], = manzana

Arrays o Matrices

- PHP también da soporte a un tipo de matriz un poco diferente, en que los números son reemplazados con cadenas de caracteres o “palabras clave” definidas por el usuario. Se llaman matrices asociativas.

```
<?php
// define una matriz
$frutas = array( 'm' => 'manzana' 'p' => 'plátano' 'i' => 'piña'
'u' => 'uva'
);
?>
```

- Las palabras clave de la matriz deben ser únicas
- Cada una de las palabras clave hace referencia a un solo valor y la relación palabra clave-valor está expresada por el símbolo =>.
- Para acceder al valor 'manzana' de la matriz, se utiliza la notación \$frutas['m']

Arrays o Matrices

Asignar valores a matrices

- Las reglas de PHP para dar nombre a las matrices son las mismas que las que se usan para las variables regulares.
 - Maneras de asignarle valores.
1. La primera es el método donde los valores son asignados uno por uno y separados por comas. ([visto anteriormente](#))
 2. La segunda manera de crear una matriz semejante es estableciendo valores individuales utilizando notaciones indexadas.

```
<?php// define una matriz
$coche [0] = 'Ferrari';
$coche [1] = 'Porsche';
$coche [2] = 'Jaguar';
$coche [3] = 'Lamborghini';
$coche [4] = 'Mercedes';
?>
```


Arrays o Matrices

Asignar valores a matrices

3. También se puede asignar automáticamente el siguiente número de índice disponible en la matriz, omitiendo los números de índice en la declaración de asignación de la matriz, como se muestra a continuación:

```
<?php
// define una matriz
$coche[] = 'Ferrari';
$coche[] = 'Lamborghini';
?>
```

4. Se puede establecer una palabra clave para cada valor y vincular ambas utilizando el conector =>

```
<?php
$datos = array( 'nombredeusuario' => 'juan', 'servidor' => '192.168.0.1');
?>
```

5. O bien asignar valores a palabras clave uno por uno, como en el siguiente ejemplo:

```
<?php
$datos['nombredeusuario'] = 'juan';
$datos['servidor'] = '192.168.0.1';
?>
```

Arrays o Matrices

Acceso a valores de matrices

- Para acceder a un valor de la matriz en el script, simplemente utiliza el nombre y el índice/palabra clave en una expresión.

```
echo "El servidor es".$datos[servidor]
```

- Para Modificar valores de matrices simplemente se define el nuevo valor en esa posición.

```
<?php
// define una matriz
$carnes = array( 'pescado', 'pollo','jamón', 'cordero');
// cambia 'jamón' por 'pavo'
$carnes[2] = 'pavo';
?>
```

- Para eliminar un elemento de la matriz, utiliza la función **unset()**, en la matriz anterior sería:

```
// elimina 'pescado'
unset($carnes[0]);
```

- Si eliminas un elemento con la función **unset()**, PHP lo declarará nulo (NULL), pero no volverá a indexar automáticamente la matriz. Si se trata de una matriz indexada numéricamente, puedes volver a indexarla, para eliminar los huecos en la secuencia, utilizando la función PHP **array_multisort()** sobre tu matriz.
- También podemos eliminar y de una matriz con **array_pop()** o **array_push()**;

Arrays o Matrices

Recuperar el tamaño de una matriz

- Saber cuántos valores contiene una matriz es una tarea importante cuando se trabaja con ellos, en especial cuando se combinan con bucles
- Se resuelve fácilmente con la función PHP **count()** o **sizeof()**, que acepta la variable de una matriz como parámetro y regresa un número entero que indica cuántos elementos contiene.

```
<?php
    $datos = array('lunes', 'martes', 'miércoles');
    // obtiene el tamaño de la matriz
    echo 'La matriz tiene ' . count($datos) . ' elementos';
?>
```

Arrays o Matrices

Matrices anidadas

- PHP también te permite combinar matrices, colocando una dentro de otra sin límite de profundidad.

```
<?php
// define matrices anidadas
$directorio = array( array(
    'nombre' => 'Raymundo Rabbit', 'tel' => '1234567',
    'correo' => 'ray@planetaconejo.in',),
    array(
        'nombre' => 'David Duck', 'tel' => '8562904',
        'correo' => 'dduck@lagodepatos.corp',),
    array(
        'nombre' => 'Omar Horse', 'tel' => '5942033',
        'correo' => 'reyomar@mercadodelgranjero.cosasdecaballos.com',));
?>
// accede al valor anidado de David Duck
echo "El número telefónico de David Duck es: " . $directorio[1] ['tel'];
?>
```

Arrays

Procesar matrices con bucles e iteradores

- En muchas ocasiones necesitamos recorrer el contenido de una matriz mediante bucles, por ejemplo, empleando el bucle **for**

```
<?php
// define una matriz
$ciudades = array('Londres', 'París', 'Madrid', 'Los Ángeles', 'Bombay', 'Yakarta');

// hace iteraciones sobre la matriz
// presenta cada valor
for ($i=0; $i<count($ciudades); $i++) { echo $ciudades[$i] . "\r\n";
}
?>
```

- En este ejemplo, el bucle for hace las iteraciones sobre la matriz \$ciudades, y presenta en pantalla cada valor encontrado. El bucle se ejecuta para cada elemento de la matriz; esta información se averigua rápidamente invocando la función count().

Arrays

Procesar matrices con bucles e iteradores

- El bucle foreach representa la manera más sencilla de realizar iteraciones sobre matrices.
- Con el bucle foreach, cada vez que se ejecuta un bucle, el elemento en uso de la matriz es asignado a una variable temporal, a nuestra elección, la cual puede ser procesada de la forma que más te plazca: mandarlo a pantalla, copiarlo a otra variable, usarlo en cálculos y demás.
- A diferencia del bucle for, foreach no utiliza un contador, porque automáticamente “sabe” la posición que ocupa dentro de la matriz en un determinado momento y se desplaza hacia adelante en forma continua hasta que alcanza el final de la matriz; en ese punto se detiene automáticamente

```
<?php
$ciudades = array('Londres', 'París', 'Madrid', 'Los Ángeles', 'Bombay', 'Yakarta');
foreach($ciudades as $c) {
    echo "$c \r\n";
}
?>
```

Arrays

Procesar matrices con bucles e iteradores

- El bucle foreach también funciona con matrices asociativas, sólo que en esos casos utiliza dos variables temporales (una para la palabra clave y otra para el valor).

```
<?php
// define una matriz
$ciudades = array(
    "Reino Unido" => "Londres", "Estados Unidos" => "Washington", "Francia" => "París",
    "India" => "Delhi"
);

// hace iteraciones sobre la matriz
// presenta cada valor

foreach($ciudades as $clave => $valor){
    echo "$valor es la capital de $clave. \r\n";
}
?>
```

Arrays

Procesar matrices con bucles e iteradores

- Vamos a realizar un **ejemplo** para entender todo lo anterior.
- Realiza el ejercicio 36 de tus cuaderno de ejercicios, el que lleva por nombre, **Promedio de clasificaciones.**

En dicho ejercicio crearemos una pequeña aplicación que acepte una matriz de valores que representaran las calificaciones numéricas individuales de un grupo de estudiantes de cierto curso, y luego calcularás varias estadísticas de resumen, tal que:

- Promedio general
- cantidad de estudiantes que se encuentran dentro del 20% superior
- y el 20% inferior dentro del grupo.

Arrays

Matrices en formularios

- **Utilizar Matrices en Formularios.**

Las matrices son muy poderosas cuando se combinan con elementos de formularios Web que dan soporte a más de un valor, como listas de selección múltiple o casillas de verificación agrupadas.

Para capturar en una matriz los datos proporcionados por el usuario, simplemente añade un juego de corchetes al elemento **'name'** del formulario para convertirlo automáticamente en una matriz de PHP cuando se envíe.

- **Ejemplo.** Examina el siguiente formulario, que contiene una lista de selección múltiple con nombres de grupos musicales:

```
<form method="post" action="matriz-formulario.php">
  Selecciona el nombre de tu artista favorito: <br />
  <select name="artistas[ ]" multiple="true">
    <option value="One republic">One republic</option>
    <option value="Black-Eyed Peas">Black-Eyed Peas</option>
    <option value="Foo Fighters">Foo Fighters</option>
  </select>
<p>
  <input type="submit" name="submit" value="Enviar" />
</form>
```

Arrays

Matrices en formularios

Pon atención al atributo **'name'** del elemento `<select>`, que tiene el nombre `artistas[]`.

```
<select name="artistas[ ]" multiple="true">
```

Esto le indica a PHP que, cuando se envíe el formulario, todos los valores seleccionados de la lista deben transformarse en elementos de una matriz. El nombre de ésta será `$_POST['artistas']`, y tendrá más o menos este aspecto, según la selección realizada, en nuestro caso el primero y último del select, fíjate en la asignación de índice dinámico.

```
Array
(
    [0] => One republic
    [1] Foo Fighters
)
```


Arrays

Matrices en formularios

- Vamos a realizar un **ejemplo** para mejorar la comprensión.
- Realiza el ejercicio 37 de tus cuaderno de ejercicios, el que lleva por nombre, **Sabores de pizzas**.

Hagamos una pequeña aplicación que presenta al usuario un formulario que contiene varios sabores de pizzas y le solicita que seleccione sus favoritas mediante casillas de verificación, imprimiremos los resultados mediante una lista sin ordenar.

Arrays

Funciones en Matrices.

- **Conversión entre cadenas de caracteres y matrices**

PHP te permite convertir una cadena de caracteres en una matriz, al tratar la cadena con separadores definidos por el usuario y asignar los segmentos resultantes a una matriz.

La función PHP que realiza esta tarea lleva el nombre de **explode()**, acepta dos argumentos: el separador y la cadena de caracteres fuente, y regresa una matriz. He aquí un ejemplo:

```
<?php
// define una cadena de caracteres
$cadena = 'policía, sastre,soldado,espía';
// convierte una cadena en matriz
// datos de salida: ('policía', 'sastre', 'soldado', 'espía')
$matriz = explode(',', $cadena);
print_r($matriz);
?>
```

- También es posible revertir el proceso, conjuntar elementos de una matriz en una sola cadena de caracteres utilizando el “pegamento” proporcionado por el usuario. La función de **implode()** permite realizar ésta operación.

Arrays

Funciones en Matrices.

- **Trabajar con rangos de números.**

Si tratas de llenar una matriz con un rango de números, la función **range()** es mejor opción que teclear manualmente cada valor. Esta función acepta dos extremos y regresa una matriz que contiene todos los números existentes entre los extremos establecidos.

He aquí un ejemplo; genera una matriz que contiene todos los valores entre 1 y 1000:

```
<?php
// define una matriz
$matriz = range(1,1000);
print_r($matriz);
?>
```

Como opción, si ya cuentas con una matriz de números y quieres calcular el mínimo y el máximo de la serie, las funciones PHP **min()** y **max()** serán de utilidad; aceptan una matriz numérica y regresan el valor menor y mayor, respectivamente, de los elementos contenidos en la matriz.

```
echo 'El mínimo es ' . min($matriz) . ' y el máximo es ' . max($matriz);
```

Arrays

Funciones en Matrices.

- **Extraer segmentos de la matriz**

PHP te permite cortar una matriz en partes pequeñas con la función `array_slice()`, que acepta tres argumentos: la matriz original, la posición del índice (offset) donde debe comenzar el corte y la cantidad de elementos que debe regresar a partir de la posición de inicio.

Observa el siguiente ejemplo:

```
<?php
// define una matriz
$arcoiris = array('violeta', 'negro', 'azul', 'verde', 'amarillo', 'naranja', 'rojo');
// extrae 3 valores centrales
// datos de salida: ('azul', 'verde', 'amarillo')
$matriz = array_slice($arcoiris, 2, 3);
print_r($matriz);
?>
```

Para extraer un segmento a partir del final de la matriz (en lugar de hacerlo desde el principio), inserta un valor negativo para la posición del índice en la función `array_slice()`, es decir quedaría,

```
$matriz = array_slice($arcoiris, -2, 3)
```

Arrays

Funciones en Matrices.

- **Añadir y eliminar elementos de la matriz**

PHP contiene cuatro funciones que te permiten añadir o eliminar elementos del principio o el final de la matriz: **array_unshift()** añade un elemento al principio; **array_shift()** elimina el primer elemento; **array_push()** añade un elemento al final; **array_pop()** elimina el último elemento de la matriz. Estas funciones no trabajan con matrices asociativas. El siguiente ejemplo los muestra en acción:

```
<?php
$filmes = array('El Rey León', 'Cars', 'Bichos');
    array_shift($filmes); // elimina el primer elemento de la matriz
    array_pop($filmes); // elimina el último elemento de la matriz
    array_push($filmes, 'Ratatouille'); // añade un elemento al final de la matriz
    array_unshift($filmes, 'Los Increíbles'); // añade un elemento al principio de la matriz
    // muestra la matriz
    // datos de salida: ('Los Increíbles', 'Cars', 'Ratatouille')
    print_r($filmes);
?>
```


Arrays

Funciones en Matrices.

- **Eliminar elementos duplicados en la matriz**

PHP te permite limpiar una matriz de valores duplicados con la función **array_unique()**, que acepta la matriz completa y regresa una nueva que sólo contiene valores únicos. Ejemplo:

```
<?php
// definir una matriz
    $duplicados = array('a', 'b', 'a', 'c', 'e', 'd', 'e');
// elimina duplicados
// datos de salida: ('a', 'b', 'c', 'e', 'd')
    $originales = array_unique($duplicados);
    print_r($originales);
?>
```

Arrays

Funciones en Matrices.

- **Ordenar aleatoriamente e invertir la matriz**

La función PHP **shuffle()** transforma el orden actual de los elementos de la matriz para darles un orden aleatorio, mientras que la función **array_reverse()** invierte el orden de sus elementos. El siguiente ejemplo lo muestra:

```
<?php
// define una matriz
$arcoiris = array('violeta', 'negro', 'azul', 'verde', 'amarillo', 'naranja', 'rojo');
// da orden aleatorio a la matriz
shuffle($arcoiris);
print_r($arcoiris);
// invierte los elementos de la matriz
// datos de salida: ('rojo', 'naranja', 'amarillo', 'verde', 'azul', 'negro', 'violeta')
$matriz = array_reverse($arcoiris);
print_r($matriz);
?>
```

Arrays

Funciones en Matrices.

- **Realizar búsquedas en la matriz**

La función `in_array()` revisa la matriz en busca de un valor específico y regresa una respuesta verdadera (true) en caso de localizarlo. He aquí un ejemplo, que busca la cadena de caracteres 'Madrid' en la matriz `$ciudades`:

```
<?php
// define una matriz

$ciudades = array('Londres', 'París', 'Madrid', 'Lisboa', 'Zurich');

// busca un valor dentro de la matriz

echo in_array('Madrid', $ciudades);

?>
```

Arrays

Funciones en Matrices.

Si en lugar de valores quieres buscar palabras clave de una matriz asociativa, PHP también puede realizar esa acción: la función **array_key_exists()** busca una coincidencia entre las palabras clave de la matriz y el término específico que se busca. El siguiente ejemplo lo ilustra:

```
<?php
// define una matriz
$ciudades = array(
    "Reino Unido" => "Londres",
    "Estados Unidos" => "Washington",
    "Francia" => "París",
    "India" => "Delhi"
);
// busca palabra clave en la matriz
echo array_key_exists('India', $ciudades);
?>
```

Arrays

Funciones en Matrices.

- **Ordenar matrices**

PHP cuenta con varias funciones integradas diseñadas para ordenar matrices de muy diversas maneras. La primera es la función **sort()**, que te permite disponer las matrices indexadas numéricamente por orden alfabético o numérico, de menor a mayor. He aquí un ejemplo:

```
<?php
// define una matriz

$datos = array(15, 81, 14, 74, 2);

// ordena y presenta la matriz

// datos de salida: (2, 14, 15, 74, 81)

sort($datos);

print_r($datos);

?>
```


Arrays

Funciones en Matrices.

Sin embargo, cuando quieres organizar una matriz asociativa, es mejor utilizar la función **asort()**, que mantiene la correlación entre palabras clave y valores mientras realiza la organización. El siguiente ejemplo lo ilustra, **ordena por los valores**:

```
<?php
// define una matriz
$perfil = array(
    "nombre" => "Susana",
    "apellido" => "De Tal",
    "sexo" => "femenino",
    "sector" => "Administración de recursos"
);
// ordena por valor
// datos de salida: ('sector' => 'Administración de recursos', 'apellido' => 'De Tal', 'sexo'
=> 'femenino', 'nombre' => 'Susana')
asort($perfil);
print_r($perfil);
?>
```

Arrays

Funciones en Matrices.

La función **ksort()** también se aplica a matrices asociativas; utiliza las palabras clave en lugar de los valores para ordenar la matriz. He aquí un ejemplo:

```
<?php
// define una matriz
$perfil = array(
    "nombre" => "Susana",
    "apellido" => "De Tal",
    "sexo" => "femenino",
    "sector" => "Administración de recursos"
);
// organiza por palabra clave
// datos de salida: ('apellido' => 'De Tal', 'nombre' => 'Susana', 'sector' =>
'Administración de recursos', 'sexo' => 'femenino')
ksort($perfil);
print_r($perfil);
?>
```

Arrays

Funciones en Matrices.

- **NOTA:**

Para invertir la secuencia ordenada que generaron `sort()`, `asort()` y `ksort()`, utiliza las funciones `rsort()`, `arsort()` y `krsort()`, respectivamente.

Arrays

Funciones en Matrices.

- **Combinar matrices**

PHP te permite combinar dos o más matrices con la función **array_merge()**, que acepta variables de una o más matrices. El siguiente ejemplo y los datos de salida muestran su uso:

```
<?php
// define matrices
    $oscuro = array('negro', 'café', 'azul');
    $claro = array('blanco', 'plateado', 'amarillo');
// combina matrices
// datos de salida:('negro', 'café', 'azul', 'blanco', 'plateado', 'amarillo')
    $colores = array_merge($oscuro, $claro);
    print_r($colores);
?>
```

Arrays

Funciones en Matrices.

- **Comparar matrices**

PHP proporciona dos funciones para comparar matrices: **array_intersect()**, que regresa los valores comunes entre dos matrices y **array_diff()** que regresa los valores de la primera matriz que no existen en la segunda. He aquí un ejemplo que ilustra ambas en acción:

```
<?php
    $naranja = array('rojo', 'amarillo');
    $verde = array('amarillo', 'azul');
    // encuentra elementos comunes
    $comunes = array_intersect($naranja, $verde); // datos de salida: ('amarillo')
    print_r($comunes);
    // encuentra elementos de la primera matriz que no están en la segunda
    $unico = array_diff($naranja, $verde); // datos de salida: ('rojo')
    print_r($unico);
?>
```

NOTA

También puedes comparar matrices con el operador de comparación de PHP (==), de manera muy similar a la comparación de variables.

Arrays

**Técnicas de búsqueda
ordenación
e
Inserción.**

Algoritmos de búsqueda

- Son algoritmos que sirven para buscar un valor dentro de un array.
 - Búsqueda secuencial:
 - Consiste en comparar cada elemento del array con el elemento que pretendemos buscar.
 - Termina cuando encontramos el elemento que buscábamos o cuando llegamos al final del array.
 - Si encuentra el elemento se devuelve la posición del array y en el caso contrario se devuelve un código de error.

Algoritmos de búsqueda

- Búsqueda binaria:
 - Se utiliza en arrays ordenados.
 - Se compara el elemento que divide el array en dos mitades, es decir el del centro, con el elemento que buscamos, si no coinciden se determina en que mitad del array se encuentra para buscar dentro de ella, descartando la otra mitad.

Algoritmos de búsqueda

○ Búsqueda binaria (continuación):

A[0]	A[1]	A[2]	A[3]
24	31	36	80

Se desea encontrar el elemento 36

1. Se calcula el elemento central del *array*:

$$centro = \frac{indice_{Bajo} + indice_{Alto}}{2} = \frac{0 + 3}{2} = 1$$

El elemento central es $a[1]=31$

2. Se compara con el elemento buscado:

Como $a[1]=31 < 36$ se descarta la primera mitad del *array* y se continua la búsqueda en la segunda mitad

36	80
----	----

3. Se calcula el elemento central del *array* de la sublista derecha:

$$centro = \frac{indice_{Bajo} + indice_{Alto}}{2} = \frac{2 + 3}{2} = 2$$

El elemento central es $a[2]=36$

4. $a[2]=36 \rightarrow$ Hemos encontrado el elemento

Nota: Las imágenes referentes a los algoritmos de ordenación y búsqueda en arrays están basadas en (Joyanes & Zahonero, 2004).

Algoritmos de búsqueda

- Búsqueda binaria (continuación):

- **PHP:**

```
array_search(variable,valor);  
$a=array(0=> 'blue', 1=> 'green');  
$b=array_search($a, 'green');
```

- **JSP:**

```
binarySearch(variable,valor);  
String[] a={'blue','green'};  
int b=binarySearch(a,'green');
```


Algoritmos de ordenación

- Son algoritmos que sirven para ordenar un array.
 - Inserción directa: se basa en la inserción de los elementos de manera ordenada en la posición que les corresponde.

A= 49,24,36,80,31

49

Comienza con el elemento 49

24	49
----	----

Se inserta el elemento 24 en la posición 0 y se mueve el elemento 49 a la posición 1

24	36	49
----	----	----

Se inserta el elemento 36 en la posición 1 y se mueve el elemento 49 a la posición 2

24	36	49	80
----	----	----	----

Se inserta el elemento 80 en la posición 3

24	31	36	49	80
----	----	----	----	----

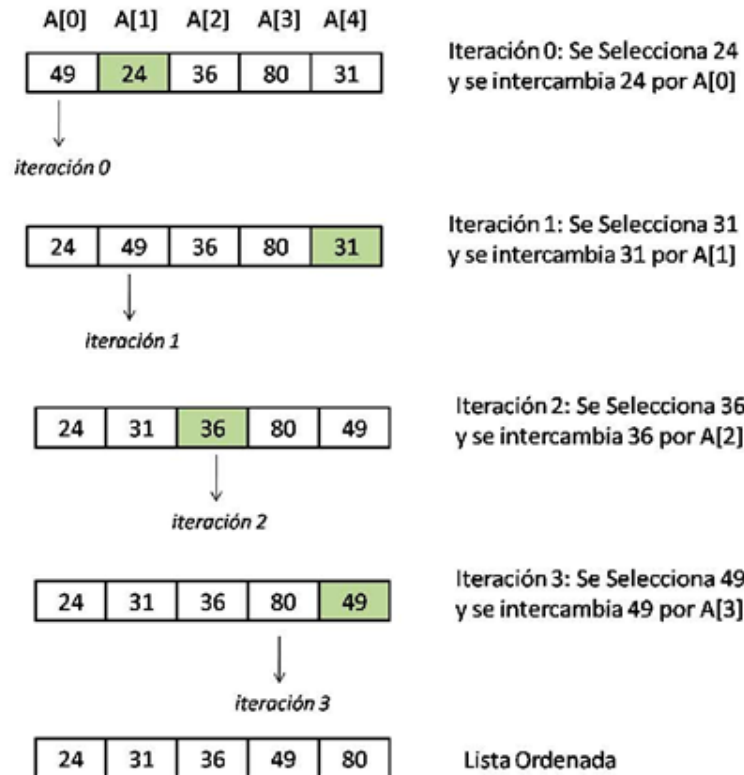
Se inserta el elemento 31 y se desplaza a la derecha la sublista derecha

Algoritmos de ordenación

- Selección directa: selecciona el elemento más pequeño de todo el array y se intercambia con el primer elemento. Se busca el siguiente más pequeño y se intercambia por el segundo y así sucesivamente hasta que quede sólo un elemento y por lo tanto la lista esté ordenada.

Algoritmos de ordenación

- Selección directa (continuación):



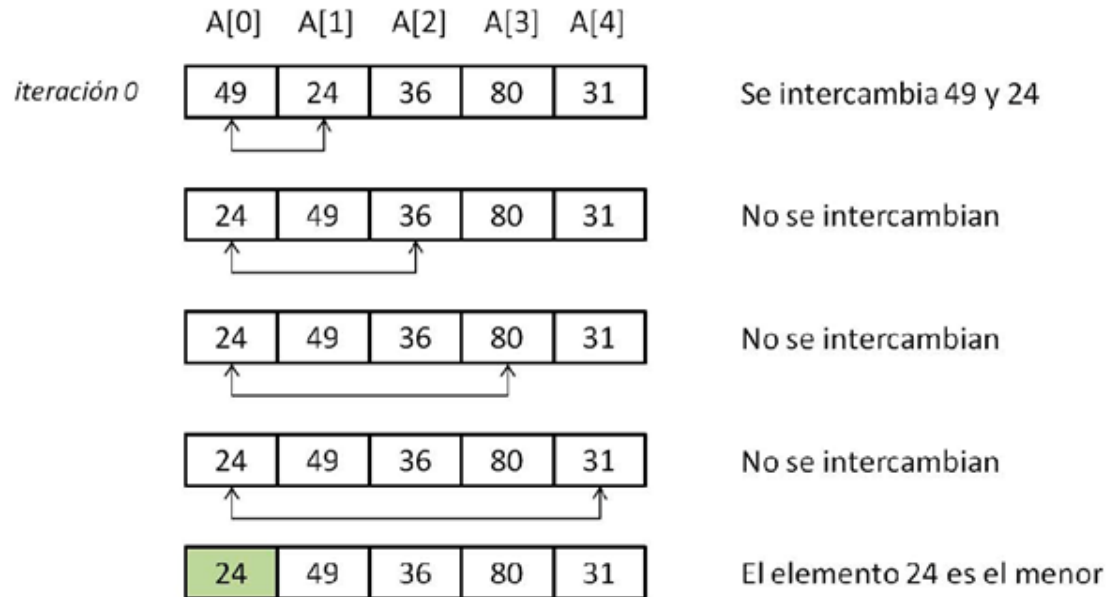
Algoritmos de ordenación

- Intercambio:

- Ordena un array de manera ascendente.
- Se basa en la lectura sucesiva del array comparando el elemento de la posición 0 de la lista con el resto, si el elemento de la posición 0 del array es mayor que el otro elemento con el que se le compara se intercambian. Una vez que hemos comparado el primer elemento con todos, se hace lo mismo con el resto de posiciones hasta conseguir ordenar el array.

Algoritmos de ordenación

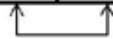
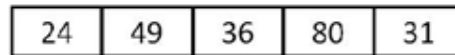
- Intercambio (continuación):



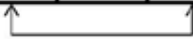
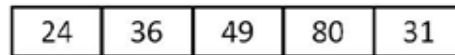
Algoritmos de ordenación

- Intercambio (continuación):

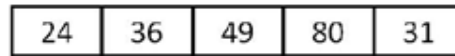
iteración 1



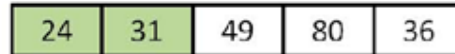
Se intercambian 49 y 36



No se intercambian



Se intercambian 31 y 36



El elemento 24 y 31 son los menores

Algoritmos de ordenación

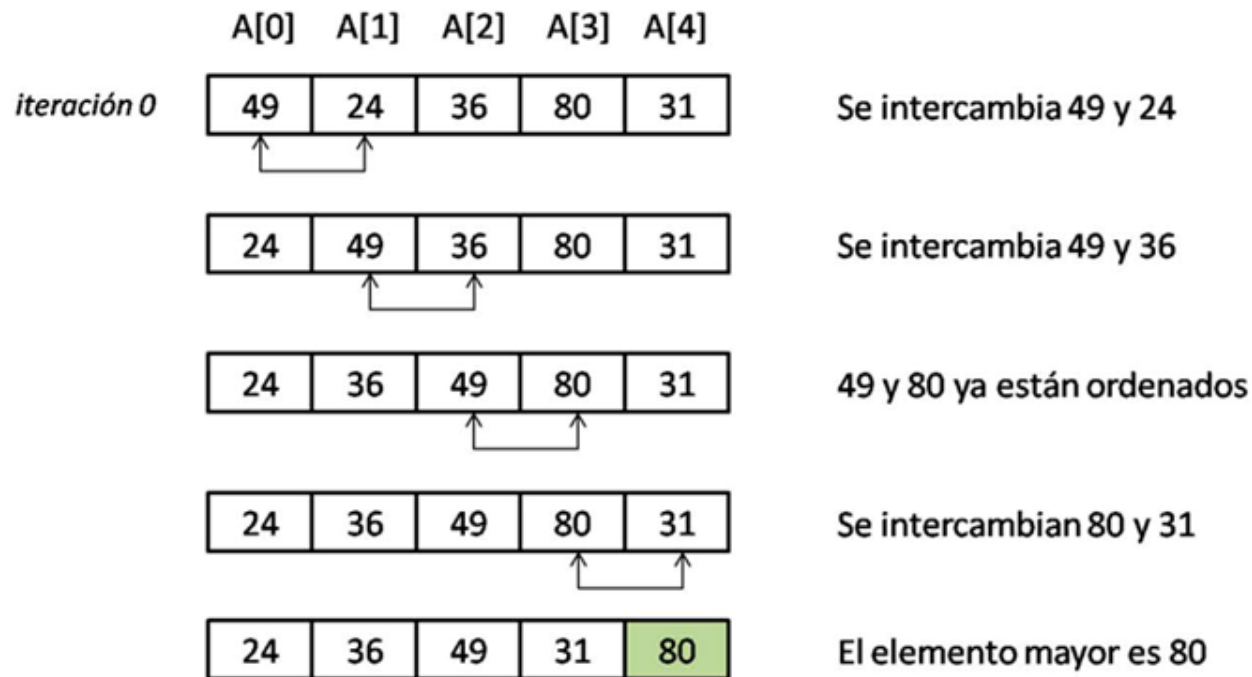
- Intercambio (continuación):

Algoritmos de ordenación

- Burbuja: consiste en el intercambio entre pares de elementos adyacentes.
 - Si un elemento no está ordenado respecto al siguiente se intercambian la posición.
 - Se realizan sucesivas iteraciones hasta que el array queda ordenado.

Algoritmos de ordenación

- Burbuja (continuación):



Algoritmos de ordenación

○ Burbuja (continuación):

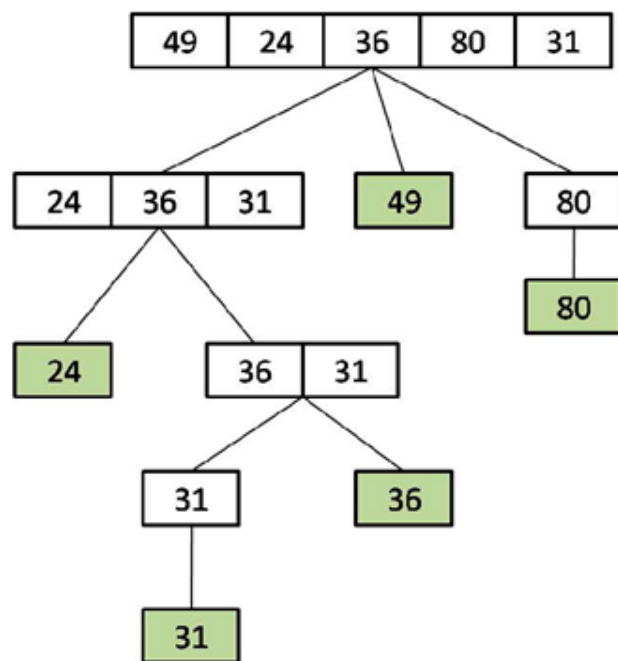


Algoritmos de ordenación

- QuickSort: divide el array en dos partes separadas por elemento central, llamado *pivote*.
 1. Elegir el pivote (ej. el primer elemento).
 2. Dividir el array de tal manera que los elementos menores que el pivote formen parte de la sublista izquierda y los que sean mayores se encuentren en la sublista derecha.
 3. Ordenar cada sublista de forma independiente aplicando este mismo algoritmo.

Algoritmos de ordenación

- QuickSort (continuación):

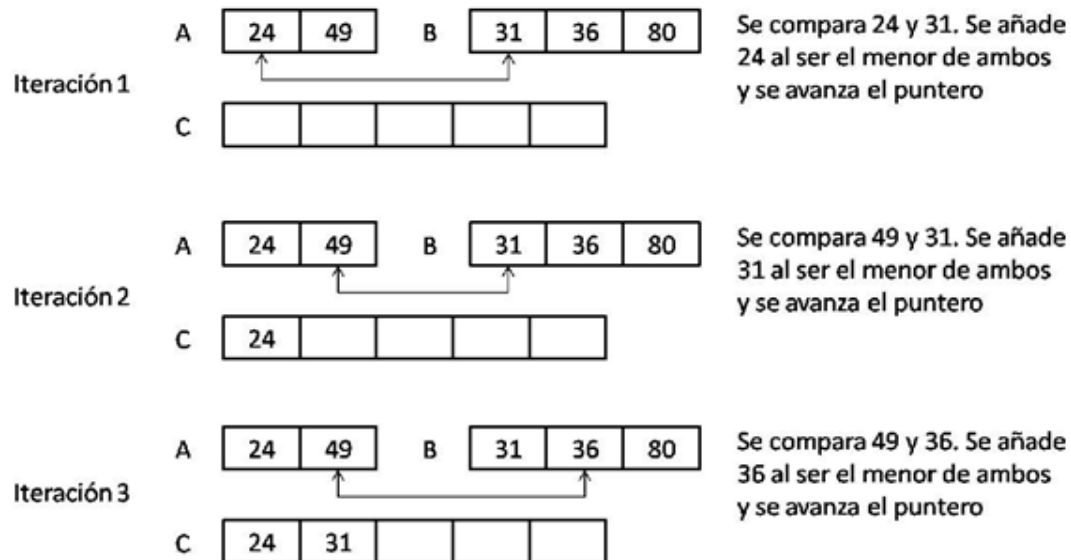


Algoritmos de ordenación

- Mergesort:
 - Parte de dos array ordenados.
 - El objetivo es obtener un array ordenado que contenga la información de ambos.
 - Se utiliza un array nuevo que tenga como tamaño la suma del tamaño de los dos arrays. Después se van comparando los elementos de cada array y se inserta en el nuevo array el menor de los dos.

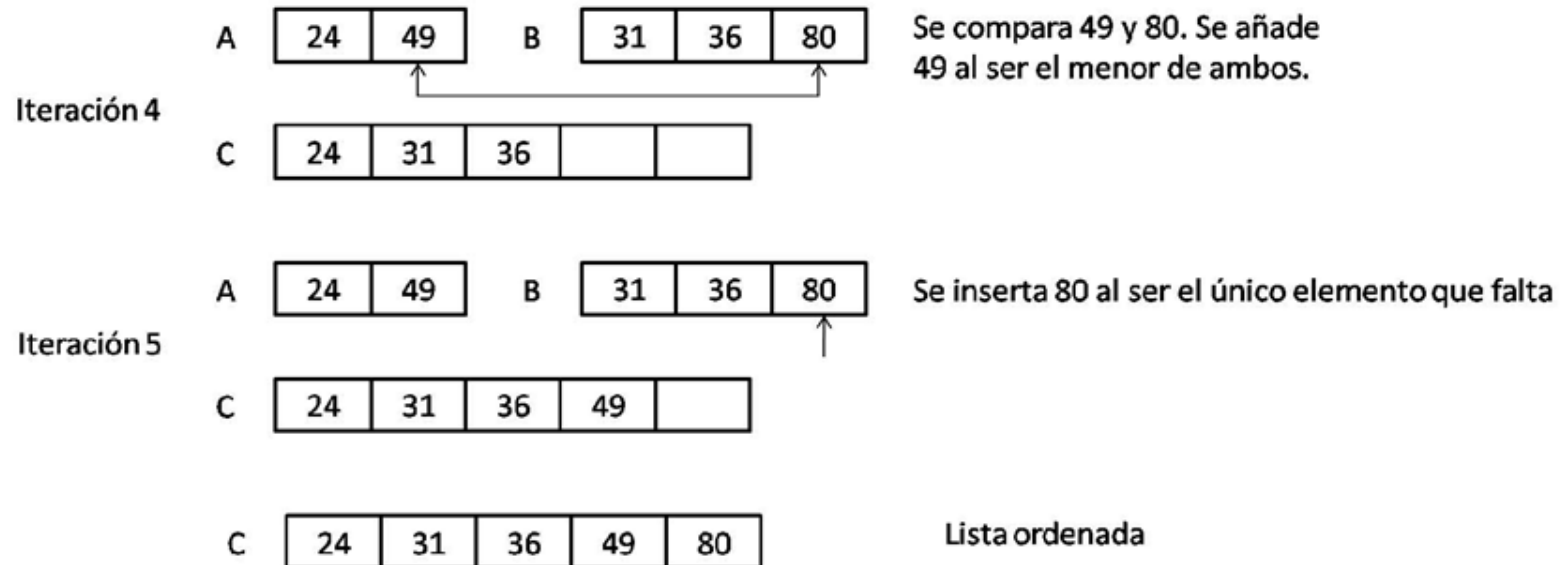
Algoritmos de ordenación

○ Mergesort (continuación):



Algoritmos de ordenación

- Mergesort (continuación):



Algoritmos de ordenación

- HeapSort: algoritmo basado en árboles binarios. Para ordenar un array de forma ascendente debemos cumplir la condición de que un nodo padre no puede ser menor a un nodo hijo.
- **PHP:**
 - Ordenar de menor a mayor: `sort(variable);`
 - Ordenar de mayor a menor: `rsort(variable);`
 - Ejemplo:

```
$a=array(0=> 21,1=> 8,2=> 9);  
rsort($a);  
sort($a);
```


Algoritmos de inserción

- Se utilizan para añadir nuevos elementos dentro de un array.
 - Si el array no está ordenado basta con añadir el elemento al final del array.
 - Si no esta ordenado hay que recorrer el array para buscar la posición exacta en la que debemos insertar el elemento y desplazar los elementos mayores que estén a la derecha.

Gracias por tu interés

Fin de la presentación