Vectores (tradicionales)

Un Array es una colección de valores. Los array pueden ser unidimensionales (vectores), bidimensionales (matrices) y multidimensionales (más de dos dimensiones)

Los arrays se utilizan ampliamente en el lenguaje PHP.

Se utiliza el delimitador [] para acceder a los diferentes elementos del vector.

Se lo puede crear al vuelo, sin tener que declararlo:

```
$dias[0]=31;
$dias[1]=28;
```

Luego de estas dos líneas, tenemos creado un vector de dos elementos, a los cuales accedemos por un subíndice.

```
echo $dias[0];  //31
echo $dias[1];  //28
```

El vector, como podemos ver, puede ir creciendo en forma dinámica, es decir que si ahora hacemos:

```
$dias[2]=31;
```

el vector ahora pasa a tener 3 componentes.

También podemos obviar el subíndice cuando asignamos los valores:

```
$dias[]=31;
$dias[]=28;
$dias[]=31;
```

Automáticamente comienza a numerarse desde cero.

Si necesitamos conocer el tamaño del vector en cualquier momento podemos llamar a la función count.

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

Si queremos imprimir todos los elementos en la página podemos hacer:

pagina1.php

```
<html>

<head>
  <title>Problema</title>
</head>

<body>

  <?php
  $nombres[] = "juan";
  $nombres[] = "pedro";
  $nombres[] = "ana";
  for ($f = 0; $f < count($nombres); $f++) {
    echo $nombres[$f];
    echo "<br>";
  }
  ?>
```

```
</body>

</html>
```

La función sizeof(<nombre del vector>) es equivalente a count

Otra forma de inicializar un vector es definirlo e inicializarlo simultáneamente:

```
$edades=array("menores","jovenes","adultos");
```

Estamos definiendo el vector edades con tres componentes, numeradas automáticamente de cero a dos.

Acotaciones

Cuando tenemos que recorrer en forma completa un vector en PHP es muy común utilizar la estructura 'foreach'. Veamos el mismo ejemplo anterior para recorrer el vector $nombres:

pagina1.php

```
<html>

<head>
  <title>Problema</title>
</head>

<body>

  <?php
  $nombres[] = "juan";
  $nombres[] = "pedro";
  $nombres[] = "ana";
  foreach ($nombres as $nombre) {
    echo $nombre;
    echo "<br>";
  }
  ?>

</body>

</html>
```

En cada repetición del 'foreach' la variable $nombre almacena una componente del vector $nombres, luego dentro del 'foreach' mostramos el contenido de la variable $nombre:

```
  foreach ($nombres as $nombre) {
    echo $nombre;
    echo "<br>";
  }
```

Creación de un archivo de texto

Una actividad fundamental es poder registrar información en el servidor (no como hemos estado haciendo hasta el momento generando sólo una página con los datos cargados)

Para la registración de datos en el servidor disponemos de dos herramientas que se complementan en muchos casos (archivos de texto y bases de datos)

En este apartado veremos como crear un archivo de texto y añadir datos al mismo.

Lo presentaremos al tema resolviendo un problema: Implementación de un libro de visitas.

Para resolver este problema plantearemos dos páginas, un formulario para realizar la carga del nombre del visitante y sus comentarios (disponemos un objeto de tipo "text" y otro de tipo "textarea"):

pagina1.html

```
<html>

<head>
  <title>Problema</title>
</head>

<body>
  <form action="pagina2.php" method="post">
    Ingrese su nombre:
    <input type="text" name="nombre">
    <br>
    Comentarios:
    <br>
    <textarea name="comentarios" rows="10" cols="40"></textarea>
    <br>
    <input type="submit" value="Registrar">
  </form>
</body>

</html>
```

Este formulario es similar a los planteados en problemas anteriores, sólo le hemos agregado al control textarea, las propiedades rows y cols que dimensionan el mismo en la pantalla:

```
    <textarea name="comentarios" rows="10" cols="40"></textarea>
```

Veamos ahora la página (pagina2.php) que graba los datos cargados en el formulario en un archivo:

```
<html>

<head>
  <title>Problema</title>
</head>

<body>
  <?php
```

```
  $ar = fopen("datos.txt", "a") or
    die("Problemas en la creacion");
  fputs($ar, $_REQUEST['nombre']);
  fputs($ar, "\n");
  fputs($ar, $_REQUEST['comentarios']);
  fputs($ar, "\n");
  fputs($ar, "-----------------------------------------------------------");
  fputs($ar, "\n");
  fclose($ar);
  echo "Los datos se cargaron correctamente.";
  ?>
</body>

</html>
```

En primer lugar creamos o abrimos el archivo de texto "datos.txt". El segundo parámetro de la función fopen indica la forma de apertura de archivo "a" (lo crea o si ya existe el archivo lo abre para añadir datos al final), "w" (crea el archivo de texto, si existe borra su contenido) y la última forma de apertura del archivo es "r" (abre el archivo para su lectura)

Como en este problema nos interesa que el archivo vaya creciendo con los datos que aportan los visitantes al sitio lo abrimos para añadir, parámetro "a".

La función fopen retorna una referencia al archivo y la almacenamos en una variable llamada $ar.

Si el archivo no se puede abrir, se ejecuta la instrucción que se encuentra luego del operador "or" en nuestro caso llamamos a la función die que finaliza la ejecución del programa PHP mostrando como mensaje el texto que le pasamos a dicha función (por ejemplo si el disco duro del servidor está lleno no se podrá crear el archivo de texto)

```
  $ar = fopen("datos.txt", "a") or
    die("Problemas en la creacion");
```

Para la grabación de datos utilizamos la función fputs que tiene dos parámetros: la referencia al archivo donde grabamos y el string a grabar.

```
  fputs($ar, $_REQUEST['nombre']);
  fputs($ar, "\n");
```

Para el salto de línea en el archivo de texto, usamos los caracteres \n.De esta forma cuando leamos el archivo de texto lo haremos línea a línea.

Cuando dejamos de trabajar con el archivo llamamos a la función fclose.

Hay que tener muy presente que el archivo se almacena en el servidor y no en la máguina de la persona que está navegando. Si nos dirigimos a la carpeta c:\xampp\htdocs encontraremos el archivo "datos.txt", tenga en cuenta que está en la máquina donde se ejecutó el script de PHP (normalmente en un servidor web). Luego veremos como leer el contenido del archivo y mostrarlo en otra página del sitio (En nuestro caso como utilizamos el equipo como cliente/servidor el archivo datos.txt se crea en la misma carpeta donde se alojan nuestras páginas php)

Lectura de un archivo de texto

Para la lectura de un archivo de texto contamos con la función fgets. Además debemos abrir el archivo para lectura.

La apertura para leer:

```
  $ar=fopen("datos.txt","r") or
    die("No se pudo abrir el archivo");
```

Para leer:

```
    $linea=fgets($ar);
```

Veamos como mostrar por pantalla el contenido del archivo "datos.txt" creado en el punto anterior:

pagina1.php

```
<html>

<head>
  <title>Problema</title>
</head>

<body>
  <?php
  $ar = fopen("datos.txt", "r") or
    die("No se pudo abrir el archivo");
  while (!feof($ar)) {
    $linea = fgets($ar);
    $lineasalto = nl2br($linea);
    echo $lineasalto;
  }
  fclose($ar);
  ?>
</body>

</html>
```

Lo primero que debemos identificar es la forma de apertura del archivo:

```
  $ar = fopen("datos.txt", "r") or
    die("No se pudo abrir el archivo");
```

El segundo parámetro de fopen es "r" es decir read (apertura para lectura), si el archivo no existe por ejemplo se ejecuta la función die que finaliza el programa mostrando el string correspondiente.

La función feof retorna true si se ha llegado al final del archivo en caso contrario retorna false. Para que se impriman todas las líneas del archivo se plantea una estructura repetitiva que se ejecuta mientras no se llegue al final de archivo (el operador lógico not en PHP es el caracter !):

```
  while (!feof($ar)) {
```

Dentro de la estructura repetitiva leemos una línea completa del archivo de texto con la función fgets:

```
    $linea = fgets($ar);
```

La variable $linea contiene una línea completa del archivo de texto, inclusive el salto de línea (\n)

Como el navegador no hace un salto de línea con este caracter, debemos convertir dicho caracter al elemento
 propia de HTML. La función que realiza esta actividad se llama nl2br (new line to br)

El resultado se almacena en una nueva variable que es la que realmente imprimimos:

```
    $lineasalto = nl2br($linea);
    echo $lineasalto;
```

Este tipo de vectores no es común a otros lenguajes, pero en PHP son de uso indispensable en distintas situaciones (ya lo empleamos cuando recuperamos información de un formulario accediendo al vector $_REQUEST que crea PHP en forma automática, cuando accedamos a datos de una base de datos también lo emplearemos etc.)

Un vector asociativo permite acceder a un elemento del vector por medio de un subíndice de tipo string.

Inicialmente uno piensa que esto nos complica las cosas, como veremos más adelante la realidad nos demuestra lo contrario.

Como ejemplo, consideremos que deseamos guardar en un vector el DNI, nombre y dirección de una persona. Empleando un vector con subíndice entero la solución sería:

```
<?php
$registro[] = "20456322";
$registro[] = "Martinez Pablo";
$registro[] = "Colon 134";
?>
```

De esta forma debemos recordar que cuando deseamos mostrar el nombre de la persona debemos acceder al subíndice 1. Esto es sencillo si tenemos un vector con tres elementos, pero que sucede si debemos almacenar 20 datos relacionados en un vector.

Un vector asociativo se define de la siguiente forma:

```
<?php
$registro['dni'] = "20456322";
$registro['nombre'] = "Martinez Pablo";
$registro['direccion'] = "Colon 134";
echo $registro['nombre'];
?>
```

Ahora vemos que para imprimir el nombre de la persona no debemos recordar una posición dentro de un vector sino un nombre de clave. Esto se hace indispensable cuando administramos un conjunto de datos grandes.

En un vector asociativo toda componente está asociada a una clave.

Otras formas de crear un vector asociativo:

```
<?php
$registro = array(
  'dni' => '20456322',
  'nombre' => 'Martinez Pablo',
  'direccion' => 'Colon 134'
);
echo $registro['dni'];
?>
```

Acotaciones

Cuando tenemos que recorrer en forma completa un vector asociativo en PHP podemos utilizar la estructura 'foreach'. Veamos un ejemplo:

pagina1.php

```
<html>

<head>
  <title>Problema</title>
</head>

<body>

  <?php
  $articulo = array(
    'codigo' => 1,
    'descripcion' => 'manzanas',
    'precio' => 30.25
  );

  foreach ($articulo as $clave => $valor) {
    echo 'Para la clave :' . $clave . " almacena el valor: " . $valor;
    echo "<br>";
  }
  ?>

</body>

</html>
```

En cada repetición del 'foreach' la variable $clave almacena el subíndice del vector y la variable $valor contiene el valor de la componente del vector:

```
  foreach ($articulo as $clave => $valor) {
    echo 'Para la clave :' . $clave . " almacena el valor: " . $valor;
    echo "<br>";
  }
```

Luego se muestra al ejecutar la aplicación:

```
Para la clave :codigo almacena el valor: 1
Para la clave :descripcion almacena el valor: manzanas
Para la clave :precio almacena el valor: 30.25
```

Podemos recorrer el vector asociativo $_REQUEST mediante un foreach e imprimir tanto la clave como el valor.

Desarrollaremos una aplicación que solicite la carga de 5 números en un formulario HTML y al presionar un botón se calcule la suma en el servidor.

pagina1.html

```
<head>
  <title>Problema</title>
</head>

<body>
  <form action="pagina2.php" method="post">
    Ingrese primer valor:
```

```
    <input type="text" name="valor1">
    <br>
    Ingrese segundo valor:
    <input type="text" name="valor2">
    <br>
    Ingrese tercer valor:
    <input type="text" name="valor3">
    <br>
    Ingrese cuarto valor:
    <input type="text" name="valor4">
    <br>
    Ingrese quinto valor:
    <input type="text" name="valor5">
    <br>
    <input type="submit">
  </form>
</body>

</html>
```

Es importante notar que al input de tipo 'submit' no le definimos la propiedad 'name', con el objetivo que no se cargue en el vector asociativo '$_REQUEST'.

Ahora la página que suma todos los valores ingresados es:

pagina2.php

```
<html>

<head>
  <title>Problema</title>
</head>

<body>
  <?php
  $suma = 0;
  foreach ($_REQUEST as $clave => $valor) {
    echo "Valor: " . $valor;
    echo "<br>";
    $suma = $suma + $valor;
  }
  echo "La suma es: " . $suma;
  ?>
</body>

</html>
```

Como vemos podemos recorrer en forma completa los 5 elementos del vector '$_REQUEST' y acceder tanto a su valor como su clave:

```
  $suma = 0;
  foreach ($_REQUEST as $clave => $valor) {
    echo "Valor: " . $valor;
    echo "<br>";
    $suma = $suma + $valor;
  }
```

Si solo queremos acceder a los valores del vector asociativo y no a sus claves, luego podemos codificar la sintaxis del foreach:

```
  foreach ($_REQUEST as $valor) {
    echo "Valor: " . $valor;
```

```
  echo "<br>";
  $suma = $suma + $valor;
}
```