Vectori în *PHP*

Spre deosebire de variabilele de tipuri scalare, o variabilă de tip vector stochează o mulțime sau o secvență de valori. Elementele unui vector pot avea tipuri scalare sau, la rândul lor, pot fi vectori (în acest caz, obținem vectori multidimensionali).

PHP permite definirea a două tipuri de vectori:

- indexați numeric
- asociativi.

Vectorii asociativi permit utilizarea unor valori mai utile drept index; acestea pot fi chiar şiruri de caractere.

În exemplele din acest curs, vom folosi vectori pentru a lucra mai uşor cu informațiile având caracter repetitiv (de exemplu, comenzile clienților).

Ce este un vector?

O variabilă scalară este o locație cu nume, în care poate fi stocată o valoare. Similar, un vector este o locație cu nume în care poate fi stocată o mulțime de valori.

Exemplu: Lista de produse oferite de site.

Stocarea informațiilor într-un vector permite:

- efectuarea aceloraşi acțiuni asupra fiecărei valori din vector, utilizând instrucțiuni de ciclare;
- tratarea unitară a întregului set de informații; astfel, toate valorile unui vector pot fi transmise unei funcții (de exemplu, *sort*()).

Valorile stocate într-un vector se numesc elemente ale vectorului. Fiecare element are un indice asociat (sau cheie), care este utilizat pentru a accesa elementul respectiv. În majoritatea limbajelor de programare, vectorii au indici numerici, care pot începe de la 0 sau 1.

PHP permite utilizarea atât a numerelor, cât şi a şirurilor de caractere ca indici ai tabloului. Tablourile pot fi indexate numeric sau se pot stabili valori cât mai sugestive sau semnificative ale cheilor (similar structurilor de tip *map*, *hash* sau dicționar din alte limbaje de programare).

Tablouri indexate numeric

Acest tip de vector este furnizat de către majoritatea limbajelor de programare. În PHP, indicii încep implicit de la 0.

Inițializarea tablourilor indexate numeric

Pentru a crea un vector ce conține cele 3 produse vândute prin intermediul site-ului exemplu, linia de cod PHP este următoarea:

```
$products = array('Tires', 'Oil', 'Spark Plugs' );
```

Ca şi *echo*, *array*() este mai degrabă o construcție a limbajului, şi nu o funcție.

Elementele unui vector pot fi copiate în alt vector utilizând operatorul de atribuire (=).

Dacă dorim ca un vector să rețină o secvență crescătoare de numere, funcția *range*() permite crearea automată a acestui vector. Funcția acceptă un parametru opțional care stabileşte distanța dintre valori.

Exemplu: Crearea unui vector cu elemente de la 1 la 10 se realizează astfel:

```
$numbers = range(1,10);
```

Pentru crearea unui vector cu numerele impare între 1 şi 10 se obține prin:

```
$odds = range(1, 10, 2);
```

Funcția poate fi folosită şi pentru caractere:

```
$letters = range('a', 'z');
```

Elementele unui vector pot fi încărcate din fişiere sau din baze de date. Există funcții care ne permit extragerea unei părți a unui vector sau reordonarea tabloului.

Accesarea conținutului tabloului

Conținutul unui vector poate fi accesat folosind numele variabilei şi un index (sau cheie), care se plasează între paranteze drepte după nume.

Folosind această notație, putem actualiza valorile elementelor din vector sau putem adăuga altele noi:

```
$products[0] = 'Fuses'; //modificare
$products[3] = 'Fuses'; //adaugare
```

Similar altor variabile PHP, tablourile nu trebuie inițializate sau create în avans, ci sunt create automat atunci când sunt folosite prima dată. Următoarele linii de cod creează acelaşi vector, pentru care anterior s-a folosit construcția *array*():

```
$products[0] = 'Tires';
$products[1] = 'Oil';
$products[2] = 'Spark Plugs';
```

Tablourile sunt redimensionate în mod dinamic pe măsură ce le sunt adăugate elemente, funcționalitate care nu există în cele mai multe limbaje de programare.

Utilizarea instrucțiunilor de ciclare pentru accesarea vectorilor

Pentru un vector indexat printr-o secvență de numere, parcurgerea se poate realiza simplu cu ajutorul unei bucle *for*:

```
for ($i = 0; $i<3; $i++) {
     echo $products[$i]." ";
}
```

Pe lângă instrucțiunea *for* se poate utiliza *foreach*, concepută special pentru lucrul cu vectori:

```
foreach ($products as $current) {
     echo $current." ";
}
```

Fiecare element este stocat în variabila $*current* şi apoi prelucrat (tipărit).

Tablouri cu indici diferiți

PHP permite definirea de vectori în care se poate asocia fiecărei valori orice cheie sau index.

Inițializarea unui vector

Următorul fragment de cod creează un vector în care numele produselor sunt chei şi prețurile acestora reprezintă valorile.

```
$prices = array('Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4);
```

Accesarea elementelor vectorului

Elementele vectorului pot fi accesate într-un mod similar vectorilor în care indicii sunt numerici (`$prices['Tires']`).

Vectorul anterior poate fi creat şi în alte moduri:

```
$prices = array('Tires'=>100 );
$prices['Oil'] = 10;
$prices['Spark Plugs'] = 4;
```

sau:

```
$prices['Tires'] = 100;
$prices['Oil'] = 10;
$prices['Spark Plugs'] = 4;
```

Utilizarea unstrucțiunilor de ciclare

Deoarece indicii nu sunt numere, nu se poate folosi un simplu contor în instrucțiunea *for* pentru a lucra cu vectorul. Se poate utiliza instrucțiunea *foreach* sau construcțiile *list*() şi *each*().

Forma instrucțiunii *foreach* este puțin diferită atunci când sunt utilizați vectori asociativi. Instrucțiunea poate fi folosită ca în exemplul anterior sau poate încorpora şi cheile:

```
foreach ($prices as $key => $value) {
      echo $key." – ".$value."<br />";
}
```

Folosind construcția *each*(), afişarea conținutului tabloului se realizează astfel:

```
while ($element = each($prices)) {
      echo $element['key'];
      echo " – ";
      echo $element['value'];
      echo "<br />";
}
```

Rezultatul execuției acestui *script* este:
http://193.226.51.37/web/curs3/each.php

Funcția *each*() returnează elementul curent dintr-un vector şi trece la următorul element. În fragmentul anterior, variabila $*element* este un vector. Locația *key* sau 0 conține cheia elementului curent, iar locațiile *value* sau 1 conțin valoarea elementului curent.

Un alt mod, mai obişnuit, utilizează construcţia *list*() pentru a împărţi vectorul într-un număr de valori. Astfel, cele două valori returnate de funcţia *each*() pot fi despărţite astfel:

```
while (list($product, $price) = each($prices)) {
      echo "$product - $price<br />";
}
```

Funcţia *list*() atribuie cele două elemente returnate de *each*() la două variabile, $*product* şi $*price*.
Observaţie: Atunci când este utilizat *each*(), elementul curent al tabloului se modifică. Dacă vectorul este folosit încă o dată în script, atunci trebuie resetat elementul curent la începutul tabloului, utilizând funcţia *reset*(). Pentru a parcurge din nou vectorul $*prices*, putem scrie:

```
reset($prices);
while ( list( $product, $price ) = each( $prices ) )
echo “$product - $price<br />”;
```

Operatori asupra tablourilor

Operator	*Nume*	*Exemplu*	*Rezultat*
+	Reuniune	$a + $b	Tabloul $b este adăugat lui $a, mai puţin elementele având chei deja existente în $a.
==	Egalitate	$a == $b	Adevărat dacă $a şi $b conţin aceleaşi elemente.
===	Identitate	$a === $b	Adevărat dacă $a şi $b conţin aceleaşi elemente, au aceleaşi tipuri şi se află în aceeaşi ordine.
!=, <>	Inegalitate	$a != $b	Adevărat dacă $a şi $b nu conţin aceleaşi elemente.
!==	Non-identitate	$a !== $b	Adevărat dacă $a şi $b nu conţin aceleaşi elemente, nu au aceleaşi tipuri sau nu se află în aceeaşi ordine.

Tablouri multidimensionale

Tablourile pot să nu fie doar simple liste de chei şi valori. O locație dintr-un vector poate reține un alt vector. Astfel, se poate crea un vector bidimensional, care poate fi gândit ca o matrice.

Presupunem că dorim să stocăm mai multe date despre produsele disponibile, folosind un vector bidimensional în care fiecare linie reprezintă un produs, iar fiecare coloană reprezintă un atribut al unui produs. Considerăm că un produs are 3 atribute: *cod*, *descriere*, *pret*. Următorul fragment de cod creează vectorul bidimensional corespunzător:

```
$products = array( array('TIR', 'Tires', 100 ),
                   array('OIL', 'Oil', 10 ),
                   array('SPK', 'Spark Plugs', 4 ) );
```

Astfel, vectorul $*products* va conține, la rândul lui, alți 3 vectori. Pentru a accesa un element al tabloului bidimensional, furnizăm doi indici: al liniei şi al coloanei. Pentru afişarea tabloului precedent, se poate proceda astfel:

```
for ($row = 0; $row < 3; $row++) {
      for ($column = 0; $column < 3; $column++) {
            echo '|'.$products[$row][$column];
      }
      echo '|<br />';
}
```

În browser vom obține: http://193.226.51.37/web/curs3/bidim.php

Putem folosi nume drept chei ale coloanelor, în loc de numere. Presupunem că dorim să stocăm mulțimea de produse anterioară folosind numele atributelor corespunzătoare drept chei pentru coloane. Fragmentul de cod este:

```
$products = array( array('Code' => 'TIR',
                         'Description' => 'Tires',
                         'Price' => 100
                        ),
                   array('Code' => 'OIL',
                         'Description' => 'Oil',
                         'Price' => 10
                         ),
                   array('Code' => 'SPK',
                         'Description' => 'Spark Plugs',
                         'Price' =>4
                         )
              );
```

În acest fel, putem avea nume semnificative pentru linii şi coloane. Pe de altă parte, se pierde posibilitatea de a folosi instrucțiunea *for* simplă pentru a parcurge coloanele.

Putem proceda astfel:

```
for ( $row = 0; $row < 3; $row++){
      echo '|'.$products[$row][ 'Code'].
           '|'.$products[$row][ 'Description'].
          '|'.$products[$row][ 'Price']. '|<br />';
}
```

Pentru a cicla şi în tablourile care utilizează indici descriptivi, putem utiliza funcțiile *each*() şi *list*() într-o instrucțiune *while*:

```
for ( $row = 0; $row < 3; $row++){
      while ( list( $key, $value ) = each( $products[$row])){
            echo "|$value";
      }
      echo '|<br />';
}
```

Putem avea vectori de dimensiune mai mare decât 2. Similar modului în care elementele tablourilor pot reține noi vectori, acei noi vectori pot reține, la rândul lor, alți vectori.

Un vector tridimensional are înălțime, lățime şi adâncime. Putem gândi un astfel de vector ca pe o „stivă” de vectori bidimensionali, în care un element este referit prin nivel, linie şi coloană.

Presupunem că produsele sunt împărțite în categorii (corespunzătoare maşinilor, camionetelor şi camioanelor) şi folosim un vector tridimensional pentru a le stoca. Tabloul tridimensional corespunzător se creează astfel:

```
$categories = array( array ( array('CAR_TIR', 'Tires', 100 ),
                             array('CAR_OIL', 'Oil', 10 ),
                             array('CAR_SPK', 'Spark Plugs', 4 )
                           ),
                     array ( array('VAN_TIR', 'Tires', 120 ),
                             array( 'VAN_OIL', 'Oil', 12 ),
                             array( 'VAN_SPK', 'Spark Plugs', 5 )
                           ),
                     array ( array('TRK_TIR', 'Tires', 150 ),
                             array('TRK_OIL', 'Oil', 15 ),
                             array('TRK_SPK', 'Spark Plugs', 6 )
                           )
                   );
```

Deoarece acest vector are doar indici numerici, se poate utiliza instrucțiunea *for* pentru a afișa conținutul său:

```
for ($layer = 0; $layer < 3; $layer++) {
      echo "Layer $layer<br />";
      for ($row = 0; $row < 3; $row++) {
            for ($column = 0; $column < 3; $column++) {
                  echo '|'.$categories[$layer][$row][$column];
            }
            echo '|<br />';
      }
}
```

Nu există vreo limită a limbajului asupra numărului de dimensiuni ale unui vector.

Sortarea tablourilor

Sortarea datelor dintr-un vector este o operație foarte utilă, care poate fi realizată cu uşurință în PHP.

Funcția *sort*()

Următorul fragment de cod are ca rezultat ordonarea alfabetică a vectorului:

```
$products = array('Tires', 'Oil', 'Spark Plugs' );
sort($products);
```

Evident, dacă vectorul conține valori numerice, acesta este ordonat crescător:

```
$prices = array( 100, 10, 4 );
sort($prices);
```

Funcția *sort*() este *case sensitive*. Această funcție acceptă un parametru opțional: putem specifica una dintre constantele *SORT_REGULAR* (valoarea implicită a parametrului), *SORT_NUMERIC* sau *SORT_STRING*. Acest parametru este util atunci când avem de comparat şiruri de caractere care pot conține numere.

Funcțiile *asort*() şi *ksort*()

În cazul în care este folosit un vector cu chei descriptive pentru a stoca articolele şi prețurile acestora, va trebui să folosim forme diferite ale funcției de sortare pentru a menține cheile şi valorile împreună, atunci când

sunt sortate. Următorul fragment de cod creează un vector ce conține cele 3 produse şi prețurile asociate lor, iar apoi sortează vectorul în ordinea crescătoare a prețurilor:

```
$prices = array('Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
asort($prices);
```

Funcția *asort*() ordonează vectorul conform valorii fiecărui element. În vector, valorile sunt prețurile, iar cheile sunt descrierile produselor. Dacă dorim să sortăm după descriere, va trebui să utilizăm funcția *ksort*(), care sortează după chei. Următorul fragment de cod are ca efect ordonarea alfabetică a cheilor vectorului:

```
$prices = array('Tires'=>100, 'Oil'=>10, 'Spark Plugs'=>4 );
ksort($prices);
```

Sortarea în ordine inversă

Cele trei funcţii prezentate anterior (*sort*, *asort*, *ksort*) se referă la sortarea în ordine crescătoare. Pentru fiecare dintre acestea există câte o funcție corespunzătoare pentru sortarea în ordine inversă: *rsort*, *arsort*, respectiv *krsort*. Acestea se utilizează într-un mod similar funcțiilor pentru ordonarea crescătoare.

Sortarea tablourilor multidimensionale

Sortarea tablourilor cu mai mult de o dimensiune, sau altfel decât în ordine alfabetică sau numerică, este mai complicată. PHP poate compara două numere sau două şiruri de caractere, dar într-un vector multidimensional un element este, al rândul lui, un vector. Două vectori nu pot fi comparați, decât dacă este creată o metodă pentru compararea lor.

Sortarea definită de utilizator

Considerăm definirea tabloului bidimensional utilizat anterior:

```
$products = array( array('TIR', 'Tires', 100 ),
                   array('OIL', 'Oil', 10 ),
                   array('SPK', 'Spark Plugs', 4 ) );
```

Dacă sortăm acest vector, în ce ordine vor apărea valorile? Am putea dori ca produsele să fie stocate în ordine alfabetică după descriere sau în ordine numerică după preț. Pentru a obține rezultatul dorit, trebuie să scriem propria funcție de comparare şi să utilizăm funcția *usort*() pentru a furniza un mod de comparare a elementelor.

Următorul exemplu sortează vectorul în ordine alfabetică pe baza celei de-a doua coloane din vector (descrierea):

```
function compare($x, $y) {
      if ($x[1] == $y[1]) {
            return 0;
      } else if ($x[1] < $y[1]) {
            return -1;
      } else {
            return 1;
      }
}
usort($products, 'compare');
```

Funcția de comparare trebuie să returneze 0 dacă $x = $y, un număr negativ dacă $x < $y, sau un număr pozitiv dacă $x > $y.

Funcția *usort*() are ca argumente vectorul care trebuie sortat şi numele funcției de comparare.

Există versiuni *uasort*() şi *uksort*(). Funcția *uasort*() este utilizată pentru vectori indexați numeric, după valoare, în cazul în care valorile sunt vectori. Similar, funcția *uksort*() se utilizează atunci când sortăm un vector indexat non-numeric după cheie, iar cheile sunt vectori.

Sortare utilizator inversă

Sortările definite de utilizator nu au variante pentru ordonarea inversă, însă putem ordona invers un vector multidimensional scriind o funcție de comparare care să întoarcă valori opuse celor pentru ordonarea crescătoare. De exemplu:

```
function reverse_compare($x, $y) {
      if ($x[2] == $y[2]) {
            return 0;
      } else if ($x[2] < $y[2]) {
            return 1;
      } else {
            return -1;
      }
}
```

Apelul `usort($products, 'reverse_compare')` va conduce la ordonarea inversă după preț.

Reordonarea vectorilor

În anumite aplicații, sunt utile şi alte funcții referitoare la ordonare. Funcția *shuffle* reordonează elementele vectorului în mod aleator. Funcția *array_reverse* furnizează o copie a vectorului cu toate elementele în ordine inversă.

Funcția *shuffle*()

Presupunem că dorim ca pe pagina principală a site-ului să apară un număr mic de produse (de exemplu, 3 articole selectate aleator). Acest lucru se poate realiza simplu dacă produsele se află într-un vector. Script-ul următor afişează 3 imagini alese aleator, reordonând vectorul aleator şi selectând primele 3 elemente:
http://193.226.51.37/web/curs3/front_page.php.

De fiecare dată când pagina este încărcată, vor apărea imagini diferite.

Funcția *array_reverse*()

Această funcție ia ca argument un vector şi creează un vector nou, cu acelaşi conținut însă în ordine inversă.

Exemplu: Crearea unui vector conținând numerele de la 10 la 1.

1) Folosind funcția *range*() se creează secvența crescătoare, care apoi se ordonează descrescător folosind *array_reverse*() sau *rsort*().

```
$numbers = range(1,10);
$numbers = array_reverse($numbers);
```

2) Creând vectorul element cu element:

```
$numbers = array();
for($i=10; $i>0; $i--) {
      array_push($numbers, $i);
}
```

În exemplul anterior, a fost creat un vector vid, în care au fost adăugate elemente cu ajutorul funcției *array_push*(). Aceasta permite adăugarea unui element la sfârşitul unui vector. Funcția care realizează acțiunea inversă, de ştergere a unui element de la sfârşitul vectorului, este *array_pop*().

3) Dacă datele constituie un domeniu de numere întregi, acesta se poate crea direct în ordine inversă astfel:

```
$numbers = range(10, 1, -1);
```

Încărcarea vectorilor din fişiere

În cursul anterior, am stocat comenzile lansate de clienți într-un fişier. Pentru a procesa respectivele comenzi, ar putea fi nevoie să reîncărcăm datele din fişier într-un vector.

Conținutul curent al comenzilor se află în fişierul: http://193.226.51.37/web/curs3/vieworders.php.

Script-ul utilizează funcția *file*() care încarcă fişierul într-un vector. Fiecare linie din fişier devine element în vector. Funcția *count*() furnizează numărul de elemente din vector.

În continuare, putem încărca fiecare secțiune a liniilor corespunzătoare comenzilor în elemente separate ale vectorului: http://193.226.51.37/web/curs3/vieworders2.txt

Fişierul este încărcat în vector şi este utilizată funcția *explode*() pentru a separa fiecare linie astfel încât aceasta să poată fi procesată (în exemplu, pusă într-o celulă separată a unui tabel).

Rezultatul este următorul: http://193.226.51.37/web/curs3/vieworders2.php

Funcția *explode* are următorul prototip:

```
array explode(string separator, string string [, int limit])
```

Parametrul opțional *limit* poate limita numărul maxim de elemente returnate.