**Višedimenzionalni nizovi i funkcionalnost nad nizovima**

**00:19:39**

Jedinica: 12 od 19

**Rezime**

* PHP podržava nizove nizova, odnosno višedimenzionalne nizove;
* Da bi se formirao višedimenzionalni niz, za članove niza inicijalizuju se takođe nizovi;
* FIFO i LIFO su dva različita odnosa prema podacima u kolekciji podataka, odnosno, dve različite vrste kolekcije podataka. FIFO – queue, LIFO – stack;
* PHP podržava nizove parova ključeva i vrednosti;
* Nizovima se može rukovati preko petlji, funkcija ili putem direktnog pristupa indeksima.

U prethodnoj lekciji smo se uglavnom služili nizovima koji sadrže jednu listu ključ/vrednost parova. Videli smo koliko korišćenje asocijativnih nizova može da bude praktičnije od numeričkih i obrnuto. Takođe smo pomenuli i multidimenzionalne nizove. Moglo bi se reći da su ovakvi nizovi zapravo nizovi nizova, jer su njihovi elementi novi nizovi, odnosno podnizovi. Ovakvih podnizova može biti više. Ukoliko niz poseduje samo jedan podniz, onda se radi o dvodimenzionalnom nizu. Dalje, ako taj podniz ima svoj podniz, niz je trodimenzionalni i tako dalje. Većem broju ljudi je teško da se snađu sa nizovima kada imaju više od tri ili četiri dimenzije.

Razmotrimo u nastavku detaljnije asocijativne nizove, a nakon toga i multidimenzionalne.

**Niz parova (asocijativni niz)**

Osim nizova u kojima svaki član ima svoju poziciju određenu indeksom niza koji predstavlja brojčanu vrednost, moguće je napraviti i niz u kome su vrednosti vezane za određene ključeve u tom nizu, odnosno stringove koji asociraju na vrednost.

Na primer, ukoliko bismo želeli da napravimo niz država, i svakoj državi u nizu pridružimo njen glavni grad, upotrebili bismo ovu vrstu niza. Pritom, države bi mogle biti ključevi, dok bi vrednosti mogli biti gradovi.

Izgradnja tog niza izgledala bi ovako:

|  |  |
| --- | --- |
| 1 | $arr = array("Serbia"=>"Belgrade", "France"=>"Paris", "England"=>"London"); |

... pri čemu vrednost sa leve strane znaka *=>*predstavlja **ključ**, a vrednost sa desne strane predstavlja **vrednost** tog ključa.

Dobijanje vrednosti određenog ključa u ovakvom nizu prilično je jednostavno:

|  |  |
| --- | --- |
| 1 | echo $arr["Serbia"]; |

Ova linija koda emitovaće vrednost za ključ „Serbia", što je u ovom slučaju Belgrade.

**Višedimenzionalni niz**

Kao što smo već pomenuli, osim jednodimenzionalnih nizova, koji su obrađeni u prethodnoj lekciji, php je u stanju da prepozna i takozvane višedimenzionalne nizove. Ovakvi nizovi korisni su za rad sa podacima koji imaju matričnu ili tabelarnu formu.

Na primer, recimo da želimo da uskladištimo podatke u vidu tabele, tako da ta tabela sadrži sledeće kolone: ID, Ime, Prezime, Matični broj.

Na primer:

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Ime** | **Prezime** | **Matični broj** |
| **1** | John | Miller | 1111111111111 |
| **2** | Peter | Andersen | 2222222222222 |
| **3** | Ana | Newman | 3333333333333 |

U tom slučaju mogli bismo iskoristiti dvodimenzionalni niz.

|  |  |
| --- | --- |
| 1  2  3  4 | $arr = array();      $arr[0] = array(1, "John", "Miller", "1111111111111");      $arr[1] = array(2, "Peter", "Andersen", "2222222222222");      $arr[2] = array(3, "Ana", "Newman", "3333333333333"); |

Ovaj niz smo mogli da ispunimo i drugačije, pri samoj inicijalizaciji:

|  |  |
| --- | --- |
| 1  2  3  4  5 | $arr = array(      array( 1, "John", "Miller", "1111111111111" ),      array( 2, "Peter", "Andersen", "2222222222222" ),      array( 3, "Ana", "Newman", "3333333333333")  ); |

Posmatrajući ovu strukturu, lako bismo došli do bilo kod elementa u nizu. Ukoliko bismo na primer, želeli da dobavimo prezime Andersen, ispratili bismo sledeću putanju:

|  |  |
| --- | --- |
| 1 | echo $arr[1][2]; |

Sada bismo na strani imali rezultat: Andersen

Takođe, kroz sva imena iz ove strukture bismo mogli proći i pomoću petlje:

|  |  |
| --- | --- |
| 1  2  3 | for($i=0; $i<count($arr); $i++){      echo $arr[$i][1]."<br>";  } |

... ili, ukoliko želimo da objavimo sve podatke iz tabele:

|  |  |
| --- | --- |
| 1  2  3 | for($i=0; $i<count($arr); $i++){      echo $arr[$i][0] . " " . $arr[$i][1] . " " . $arr[$i][2] . " " . $arr[$i][3] ."<br>";  } |

Ova rešenja su praktična kada imamo fiksan broj članova podniza (u ovom slučaju to su kolone naše tabele), međutim, ne i ukoliko želimo da rukujemo nekim dinamičkim podacima, čija količina nije unapred poznata. Na primer, program koji rukuje slikama, u kome korisnik može da učita sliku bilo kog formata. U tom slučaju, morali bismo da radimo sa nizovima nepoznate veličine i morali bismo da upotrebimo nešto drugačiji metod za iteraciju kroz njegove članove.

Npr. matrica koja ima vrednosti:

1 2 3 4  
5 6 7 8  
9 0 1 2

Ova matrica bi u formi niza izgledala ovako:

<?php

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | <?php  $arr = array(  array( 1, 2, 3, 4 ),  array( 5, 6, 7, 8 ),  array( 9, 0, 1, 2 )  );  ?> |

... i lako bi bilo napraviti iteraciju kroz nju, izvedenu iz malopređašnjeg primera.

Ali, ukoliko zamislimo da smo, umesto ove poznate strukture, dobili samo jednu promenljivu $arr, čije podatke ne znamo (osim da je dvodimenzionalni niz), morali bismo da pribegnemo drugoj tehnici.

|  |  |
| --- | --- |
| 1  2  3  4  5 | for($sp=0; $sp<count($arr); $sp++){      for($up=0; $up<count($arr[$sp]); $up++)          echo $arr[$sp][$up] . " ";              echo "<br>";  } |

Ali, ni ovo nije dovoljno da bi se sačuvali podaci o nekoj slici, jer slika na svakoj koordinati poseduje određeni set podataka (o boji). S obzirom na to da se boja najčešće sastoji od tri vrednosti (ukoliko se radi o RGB modelu), za svaku tačku moraćemo da odvojimo još jedan niz od tri člana.

Postoje, naravno, i druge, mnogo efikasnije metode da se rukuje ovakvom vrstom podataka (npr. objekti).

Da bismo ispunili pomenuti zahtev, potrebno je malo izmeniti dosadašnji kod. Pre svega, recimo da niz koji smo dobili parsiranjem neke slike izgleda ovako:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | $arr = array(              array(                  array(0,0,0),                  array(255,255,255),                  array(0,0,0),                  array(255,255,255)              ),              array(                  array(255,255,255),                  array(0,0,0),                  array(255,255,255),                  array(0,0,0)              ),              array(                  array(0,0,0),                  array(255,255,255),                  array(0,0,0),                  array(255,255,255)              )  ); |

 tako da je svaki podniz glavnog niza jedan red, a svaki podniz u jednom redu jedna tačka.

Za iteraciju kroz ovako serijalizovane podatke, mogli bismo iskoristiti sledeći kod:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | for( $sp = 0; $sp < sizeof( $arr ); $sp++ )      {      for( $up = 0; $up < sizeof( $arr[ $sp ]); $up++ )          for( $up1 = 0; $up1 < sizeof( $arr[ $sp ][ $up ]); $up1++ )              echo $arr[ $sp ][ $up ][ $up1 ] . " ";          echo "<br>";      } |

Jedna od olakšica koju je moguće koristiti prilikom rada sa višedimenzionalnim nizovima je **foreach**petlja. Ova petlja sama vodi računa o veličini i pozicijama članova niza prilikom iteracije, pa je dobra za neke jednostavnije operacije. Sa druge strane, ovakav pristup smanjuje kontrolu nad samim članovima niza. Foreach petlja će biti detaljnije opisana u jednoj od narednih lekcija.

Ovako bi izgledalo kretanje kroz melopređašnji niz, korišćenjem foreach petlje:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <?php    $arr = array(  array(array(0,0,0),array(255,255,255),array(0,0,0),array(255,255,255)),  array(array(255,255,255),array(0,0,0),array(255,255,255),array(0,0,0)),  array(array(0,0,0),array(255,255,255),array(0,0,0),array(255,255,255))  );  foreach($arr as $row){        foreach($row as $point){              foreach($point as $color){                    echo $color . " ";              }        }  echo "<br>";  } |

**Za koju vrstu podataka je pogodno korišćenje višedimenzionalnih nizova?**

 Ovakvi nizovi korisni su za rad sa podacima koji imaju matričnu ili tabelarnu formu.

 Ovakvi nizovi su pogodni za podatke izražene u vidu brojeva

 Ovakvi nizovi su pogodni za rad sa boolean vrednostima

**Napredne funkcije nad nizovima**

Za prikaz članova niza, bez kretanja kroz pojedinačne članove (kako je rađeno u prethodnim primerima) moguće je koristiti funkciju print\_r. Ova funkcija razlaže niz, prikazujući njegove ključeve ili brojeve indeksa i vrednosti koje se nalaze ispod njih.

Na primer:

|  |  |
| --- | --- |
| 1  2 | $arr = array("Serbia"=>"Belgrade", "France"=>"Paris", "England"=>"London");  print\_r($arr); |

imaće sledeći rezultat:

Array ( [Serbia] => Belgrade [France] => Paris [England] => London )

Korišćenje ove funkcije može biti veoma korisno tokom razvoja aplikacije, ali nije preporučljivo koristiti je u samoj produkciji, već elemente niza na izlazu treba isparsirati na pogodniji način (najčešće kroz neku vrstu petlje).

**Array\_walk()**

Funkcija array\_walk() služi za provlačenje elemenata niza kroz naznačenu funkciju. Ključevi i vrednosti elementa u nizu predstavljaju argumente funkcije. Ukoliko promene koje izazove funkcija na ključevima i vrednostima želimo odmah da sačuvamo na istom nizu, možemo iskoristiti operator & kako bismo ukazali na memorijsku adresu.

Primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | $arr = array("Serbia"=>"Belgrade", "France"=>"Paris", "England"=>"London");    array\_walk($arr, 'arrFunc', ' is capital city of state which name is ');    function arrFunc($value, $key, $p){      echo $value . $p . $key . ".<br>";  } |

Funkcija prolazi kroz sve elemente niza $arr i nad svakim poziva funkciju arrFunc. Funkcija prihvata ključ i vrednost elementa i dodatni parametar u vidu stringa, koji ima vrednost: „je glavni grad države koja se zove”. Nakon što funkcija obradi element, prelazi na sledeći, tako da je rezultat na strani:

Belgrade is capital city of state which name is Serbia.  
Paris is capital city of state which name is France.  
London is capital city of state which name is England.

**Array\_fill()**

Ukoliko želimo da formiramo niz ispunjen nekom statičkom vrednošću, možemo upotrebiti ovu funkciju, pri čemu je prvi parametar – početni indeks niza, drugi parametar – broj ubačenih članova, a poslednji parametar – vrednost koja se ubacuje.

|  |  |
| --- | --- |
| 1  2 | $x = array\_fill( 0, 10, "hello" );  print\_r( $x ); |

Ovaj kod vraća:

Array ( [0] => hello [1] => hello [2] => hello [3] => hello [4] => hello [5] => hello [6] => hello [7] => hello [8] => hello [9] => hello )

Na ovaj način moguće je jednostavno kopirati više nizova u drugi niz, odnosno, napraviti višedimenzionalni niz, s obzirom na to da ova funkcija kao parametar može primiti i sam niz.

**Array\_flip()**

Menja položaje ključeva i vrednosti. Ukoliko je niz sastavljen od ključeva i vrednosti, napraviće zamenu ključeva i vrednosti, a ukoliko niz nema ključeve, već samo indekse, zameniće indekse članova za vrednosti članova.

Kod:

|  |  |
| --- | --- |
| 1  2  3 | $arr = array("Serbia" => "Belgrade", "France" => "Paris", "England" => "London");  $x = array\_flip($arr);  print\_r($x); |

emituje:

Array ( [Belgrade] => Serbia [Paris] => France [London] => England )

**Array\_pop()**

Da bismo razumeli ovu i sledeću funkciju nad nizovima, potrebno je prvo da razumemo dva načina za pristup bilo kakvim kolekcijama podataka u programiranju – FIFO i LIFO. FIFO, što je skraćenica za First In First Out (prvi unutra, prvi napolje), podrazumeva da element koji je prvi pristupio kolekciji prvi i napušta kolekciju podataka. Dok LIFO, Last In First Out (poslednji unutra, prvi napolje), podrazumeva da element koji je poslednji pristupio kolekciji prvi napusti kolekciju.

Često se ova dva pristupa porede i sa radnim stolom i redom u pošti.

* + *Radni sto (LIFO):*Ukoliko stavimo gomilu papira na sto, onaj koji smo stavili poslednji biće na vrhu gomile i prvo ćemo njega uzeti.
  + *Red u pošti (FIFO):*Onaj ko je poslednji stao u red poslednji će doći do šaltera.

Identično je i u programiranju: stack (sto) i queue (pošta). Obično se (generalno u programiranju) za ova dva tipa punjenja kolekcije podataka vezuju i dve funkcije. Za stack to su pop i push, a za queue, dequeue i enqueue.

Array\_pop() će, dakle, izvući i obrisati poslednji element iz niza, ali i postaviti marker na prethodnu poziciju.

Dakle, sledeći kod će emitovati Vašington kao rezultat:

|  |  |
| --- | --- |
| 1  2 | $arr = array( "Belgrade", "Paris", "London", "Washington" );  echo array\_pop( $arr ); |

ali će sledeći kod emitovati: *WashingtonLondonParisBelgrade*

|  |  |
| --- | --- |
| 1  2  3  4  5 | $arr = array( "Belgrade", "Paris", "London", "Washington" );  echo array\_pop( $arr );  echo array\_pop( $arr );  echo array\_pop( $arr );  echo array\_pop( $arr ); |

Da bismo po istom principu dodali element u niz, koristimo funkciju array\_push().

**Array\_push()**

|  |  |
| --- | --- |
| 1  2  3 | $arr = array("Belgrade", "Paris", "London");  array\_push( $arr, "Washington" );  print\_r( $arr ); |

Rezultat je:

Array ( [0] => Belgrade [1] => Paris [2] => London [3] => Washington )

Ova funkcija daje rezultat identičan onome koji bismo dobili kada bismo vrednost uneli direktno, naredbom:

$arr[] = "Washington";

Ali, korišćenje array\_push() funkcije daje nešto brži rezultat, zbog čega je bolja pri radu sa većim količinama podataka.

**list()**

Dodeljuje vrednosti niza prosleđenoj listi varijabli.

|  |  |
| --- | --- |
| 1  2  3 | $capitals = array("Belgrade", "Paris", "London");  list($Serbia, $France, $England) = $capitals;  echo $France; |

Ovakav pristup dobar je za malu količinu statičkih podataka.

Rezultat koda će biti:

*Paris*

**Vežba 1**

***Problem:***

Dat je sledeći niz:

|  |  |
| --- | --- |
| 1 | $table = array("format" => array(7, 7),"positions" => array(array(3, 5),array(1, 4),array(4, 6),array(3, 7))); |

Potrebno je uz pomoć petlji proći kroz odgovarajuće nizove, tako da se kreira tabela dimenzija navedenih u podnizu format, niza table.

Potrebno je nacrtati tabelu, tako da sva prazna polja budu prikazana oznakom 0, a sva polja koja se nalaze u podnizu positions niza table, budu prikazana oznakom X. Za zadati niz, izlaz treba da izgleda ovako:

0000000  
0000000  
0000000  
X000000  
00X0000  
000X000  
00X0000

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <?php  $table = array("format" => array(7, 7),"positions" => array(array(3, 5),array(1, 4),array(4, 6),array(3, 7)));    for($fh=1; $fh<=$table['format'][1]; $fh++){      for($fw=1; $fw<=$table['format'][0]; $fw++){          $pointExists = false;          foreach($table['positions'] as $pos){              if($pos[0] == $fh && $pos[1] == $fw){                  echo 'X';                  $pointExists = true;              }          }          if(!$pointExists) echo '0';      }      echo '<br>';  }  ?> |

Informacije o tabeli koju treba kreirati nalaze se u nizu $table. Ovaj niz poseduje dva elementa. Prvi element je:

|  |  |
| --- | --- |
| 1 | "format" => array(7, 7) |

i njime se definišu dimenzije tabele, dok je drugi:

|  |  |
| --- | --- |
| 1 | "positions" => array(array(3, 5),array(1, 4),array(4, 6),array(3, 7)) |

i njime se definišu tačke u kojima će biti prikazan znak X.  
  
Da bismo na strani generisali tabelu, potrebno je da iskoristimo dve for petlje, kako bismo rešili prikazivanje kolona i redova:

|  |  |
| --- | --- |
| 1  2 | for($fh=1; $fh<=$table['format'][1]; $fh++){      for($fw=1; $fw<=$table['format'][0]; $fw++){ |

Promenljiva $pointExists na početku iteracije ima boolean vrednost false koju kasnije može promeniti, ukoliko postoji poklapanje sa definisanim tačkama i brojačima petlji.  
  
Zatim, upravo takvo poklapanje i proveravamo kroz foreach petlju:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | foreach($table['positions'] as $pos){              if($pos[0] == $fw && $pos[1] == $fh){                  echo 'X';                  $pointExists = true;              }          } |

Ukoliko je poklapanje utvrđeno, i promenljiva $pointExists menja svoju vrednost na true. Ovo je značajno jer samo ukoliko tačka poklapanja ne postoji, mi želimo da ispišemo na strani „0“.

**Vežba 2**

***Problem:***

Dat je sledeći niz:

|  |  |
| --- | --- |
| 1 | $population = array("London" => 7556900,"Belgrade" => 1500000,"New York" => 8406000, "Paris" => 11836970); |

Potrebno je napraviti program koji će na osnovu podataka o broju stanovnika formirati primitivni grafikon i u njemu prikazati broj stanovnika u skali od 1 do 10 prema postotcima. Postotke formirati tako da grad sa najvećim brojem stanovnika bude 100%.

U grafikonu treba delove koji su obuhvaćeni skalom prikazati oznakom #, dok delove koji ne pokrivaju vrednost ne treba prikazati (ove oznake su proizvoljne).

Na dnu liste treba vertikalno prikazati nazive gradova.

Konačan izgled primera:

               #      
               #      
          #   #      
#        #   #    
#        #   #    
#        #   #    
#        #   #    
#        #   #    
#   #   #   #    
#   #   #   #    
#   #   #   #    
L   B   N   P    
o    e   e   a    
n    l    w   r    
d    g        i    
o    r   Y   s    
n   a   o        
     d    r        
     e    k

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | <?php  $population = array("London" => 7556900, "Belgrade" => 1500000, "New York" => 8406000, "Paris" => 11836970);  $max = 0;  $longestCityName = 0;  foreach($population as $k => $v)      {      if($v > $max)          $max = $v;      if(strlen( $k ) > $longestCityName)          $longestCityName = strlen( $k );      }  $percent = 100 / $max;    for($i = 10; $i >= 0; $i--)  {  foreach($population as $k => $v)      {          $currentPercent = ceil(($percent \* $v)/10);               if($currentPercent >= $i)                          echo "#" . "&nbsp;&nbsp;&nbsp;";               else                          echo "<span style='color:white;'>#</span>" . "&nbsp;&nbsp;&nbsp;";      }  echo "<br>";  }  for($i = 0; $i < $longestCityName; $i++)  {      foreach($population as $k => $v)          {              if(strlen( $k ) > $i)              {                  $cityArr = str\_split( $k );                  echo $cityArr[ $i ] . "&nbsp;&nbsp;&nbsp;";              }              else                  echo "&nbsp;&nbsp;&nbsp;&nbsp;";          }          echo "<br>";  }  ?> |

Kada pogledamo zahteve zadatka i kod koji predstavlja njegovo rešenje, uočavamo da sem funkcije ceil(), ne postoje delovi sintakse sa kojima se već nismo susretali i koje nismo obradili. Zato ćemo se ovde osvrnuti upravo na funkciju ceil(). Funkcija ceil() zaokružuje dati broj na najbližu višu celobrojnu vrednost, ako je to potrebno. Ukoliko funkciji prosledimo ceo broj, funkcija na mesto poziva vraća upravo taj broj (neizmenjen). Funkcija koristi jedan parametar, koji predstavlja broj koji treba zaokružiti. Da bismo broj zaokružili na prvu manju celobrojnu vrednost, koristili bismo funkciju floor().

**Da li je moguće napraviti niz u kome su vrednosti vezane za određene ključeve u tom nizu?**

 DA

 NE