**Rad sa nizovima**

Jedinica: 11 od 19

**Rezime**

* Podrazumevani način indeksiranja u PHP-u, ako se radi o numeričkom indeksiranju, počinje od 0;
* Niz može biti kreiran i korišćenjem funkcije *array*();
* count(), sizeof() vraćaju dužinu niza;
* sort(), asort(), rsort(), ksort(), krsort() funkcije se koriste za sortiranje nizova;
* shuffle() funkcija se koristi za proizvoljno raspoređivanje elemenata niza;
* explode(), implode() funkcije se koriste za konvertovanje stringa u niz i obrnuto;
* array\_diff( $niz1, $niz2, ... ) funkcija identifikuje elemente u navedenim nizovima koji se razlikuju;
* array\_sum() funkcija sabira vrednosti koje su elementi niza;
* array\_unique() – Pomoću ove funkcije moguće je ukloniti iste elemente jednog niza;
* array\_merge($imeNiza1,  $imeNiza2) funkcija vam omogućava da dva niza spojite u jedan;
* U PHP-u je moguće kreirati novi niz kao podskup nekog već postojećeg pomoću funkcije array\_slice($imeNiza, pozicija, dužina).

Kroz dosadašnje lekcije smo se sustretali sa nizovima i osnovom sintakse za njihovo kreiranje. Ovde ćemo se detaljnije upoznati sa sintaksom za upravljanje  nizovima, kao i sa različitim tipovima nizova koji su dostupni u PHP-u.

Nizovi predstavljaju strukture podataka koje su u stanju da skladište jednu ili više vrednosti pod jednom promenljivom. Na primer, ukoliko bismo želeli da na neki način skladištimo 3 marke automobila, mogli bismo da kreiramo tri različite promenljive:

|  |  |
| --- | --- |
| 1  2  3 | $car1 = "BMW";  $car2 = "Audi";  $car3 = "Fiat"; |

Znači li to da bi za potrebe skladištenja 100 naziva marki automobila trebalo kreirati 100 različitih promenljivih? Ovo nije naročito elegantno rešenje, zbog čega treba iskoristiti prednosti koje nam nude nizovi.

Pogledajmo sada kako bismo vrednosti promenljivih koje smo kreirali iznad mogli smestiti u niz:

|  |  |
| --- | --- |
| 1 | $cars = array("BMW", "Audi", "Fiat"); |

Upravo smo kreirali niz koji je smešten u promenljivu $cars i koji ima tri elementa. Isti efekat bismo mogli postići i sledećom sintaksom:

|  |  |
| --- | --- |
| 1  2  3 | $cars[0] = "BMW";  $cars[1] = "Audi";  $cars[2] = "Fiat"; |

U nastavku lekcije ćemo detaljnije rezmotriti ovu sintaksu.

PHP razlikuje tri tipa za nizove:

* 1. **Numerički nizovi,**
  2. **Asocijativni nizovi,**
  3. **Multidimenzionalni nizovi.**

**Numerički i asocijativni nizovi**

Za razliku od promenljivih, koje čuvaju pojedinačne vrednosti, nizovi se koriste za čuvanje skupa ili sekvence vrednosti. Svaki niz se sastoji od dva dela, **indeksa** i **vrednosti**. Indeks niza se koristi za identifikaciju i pristup elementu.

[Niz](https://www.link-elearning.com/linkdl/opisPojma.php?id=139118) u PHP-u može da bude kreiran na nekoliko načina. Jedan od njih je da se svakom elementu pojedinačno dodeli vrednost. Na primer:

|  |  |
| --- | --- |
| 1  2  3 | $user[0] = "Thomas M.";  $user[1] = "Peter L.";  $user[2] = "George N."; |

Na ovaj način je definisan niz *$user* koji ima tri elementa numerisana brojevima, počevši od broja 0. Nizovi indeksirani brojevima nazivaju se numerički nizovi.

Elementi nizova mogu da budu indeksirani i pomoću stringova. Ovako indeksirani nizovi se nazivaju asocijativni nizovi. Na primer:

|  |  |
| --- | --- |
| 1  2  3 | $city['BG'] = "Belgrade";  $city['NY'] = "New York";  $city['LA'] = "Los Angeles"; |

Ukoliko koristite sledeće naredbe:

|  |  |
| --- | --- |
| 1  2  3 | $city[] = "Belgrade";  $city[] = "New York";  $city[] = "Los Angeles"; |

tada će biti kreiran niz od tri elementa, pri čemu su elementi numerisani automatski brojevima počevši od broja 0. Indeksiranje brojevima počevši od 0 je podrazumevan način indeksiranja u PHP-u, ukoliko eksplicitno ne navedete drugačije.

Niz može biti kreiran i korišćenjem funkcije *array*(), na jedan od sledećih načina:

|  |  |
| --- | --- |
| 1  2  3 | $city = array("Belgrade", "New York", "Los Angeles");  $city = array(12 => "Belgrade", "New York", "Los Angeles");  $city = array(“BG” => "Belgrade", "NY" => "New York","LA" => "Los Angeles"); |

Prva naredba kreira niz od tri elementa indeksiranih brojevima 0, 1 i 2. Druga naredba kreira niz čiji su elementi indeksirani brojevima 12, 13 i 14. Trećom naredbom se kreira niz i elementima se dodeljuju stringovi za indekse.

Svi do sada navedeni načini kreiranja nizova podrazumevaju da programer mora da navede vrednosti svakog elementa niza. Ponekad je to potpuno nepraktično i nepotrebno. Na primer, ukoliko želimo da u jednom nizu čuvamo brojeve od 1901 do 2000, dovoljno je da koristimo funkciju *range()* na sledeći način:

|  |  |
| --- | --- |
| 1 | $twentiethCentury = range(1901, 2000); |

Ovom naredbom kreiran je niz od sto elemenata čije su vrednosti redom brojevi od 1901 do 2000. Ovi elementi su indeksirani automatski indeksima od 0 do 99.

Kada je niz jednom kreiran, njegovim elementima se pristupa preko njihovih indeksa. Svaki element niza se koristi u kodu kao i bilo koja druga promenljiva.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | <html>  <head>    <title> Arrays </title>  </head>  <body>  <p>  <?php  $colors = array( 'red', 'green', 'blue' );  echo "First color in array " . $colors[0] . "<br>";  echo "Second color in array " . $colors[1] . "<br>";  echo "Third color in array " . $colors[2] . "<br>";  ?>  </p>  </body>  </html> |

Promenljiva *$colors[0]*odgovara prvom elementu niza i ima vrednost *'red'*. Izlaz prethodnog koda će biti

*First color in array is red  
    Second color in array is green  
    Third color in array is blue*

Indeksi su automatski dodeljeni elementima niza prilikom njegovog kreiranja, pri čemu prvi element ima indeks 0.

Nekada je potrebno, nakon što je niz kreiran, ukloniti neki od njegovih elemenata. Ovo se postiže naredbom *unset().* Na primer, *unset($user[2])* će ukloniti drugi element iz niza kreiranog u prvom primeru. Niz klijent će se tada sastojati samo od elemenata *$user[1]* i *$user[3].* Obratite pažnju da se uklanjanjem jednog elementa niza ne menjaju indeksi ostalih.

**Multidimenzionalni nizovi**

Element niza može biti bilo koji tip podatka. To znači da element niza može biti broj, string, boolean vrednost, ali i drugi niz. Kada jedan niz kao svoj element sadrži drugi niz, onda takav niz nazivamo multidimenzionalni niz.

Primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | $cars = array(    array("BMW",'525',2015),    array("Audi",'A8',2014),    array("Fiat",'500L',2015)  );    foreach($cars as $car){    echo "Brand: " . $car[0] . ", model: " . $car[1] . ", age: " . $car[2] . ".</br>";  } |

Miltidimenzionalne nizove ćemo detaljno obraditi u narednoj lekciji.

**Funkcije za rad sa nizovima**

PHP sadrži veliki broj funkcija za rad sa nizovima. Korišćenje ovih funkcija olakšava rad sa podacima i pisanje koda. Takođe, i većina običnih funkcija kada se primeni na niz daje drugačije rezultate nego kada se primeni na običnu promenljivu.

***count(), sizeof()***

Obe funkcije kao vrednost vraćaju broj elemenata niza. Primer:

|  |  |
| --- | --- |
| 1  2  3  4  5 | $my\_array = array("red","green","blue","yellow","purple");    echo(sizeOf($my\_array));  echo "<br/>";  echo(count($my\_array)); |

***sort(), asort(), rsort(), ksort(), krsort()***

Ove funkcije se koriste za sortiranje elemenata niza. PHP originalno čuva elemente niza u onom redosledu u kome su i kreirani, ali je zato moguće pomoću funkcija sortiranja urediti niz u redosledu u kome želite. Funkcija *sort()* uređuje niz u rastućem redosledu u odnosu na vrednosti elemenata, pri čemu se vrši ponovno indeksiranje. Pogledajte sledeći niz:

     $city[0] = "Belgrade";  
     $city[1] = "New York";  
     $city[2] = "Los Angeles";

Nakon naredbe *sort( $city ),* niz *$city* će imati sledeće elemente

         $city[0] = "Belgrade";  
     $city[1] = "Los Angeles"  
     $city[2] = "New York";

Elementi su sortirani po abecedi, pa su tako gradovi Los Angeles i New York zamenili mesta.

Ukoliko želite da sortirate asocijativni niz, a da pri tome svaki element zadrži postojeći indeks, koristite funkciju *asort().* Funkcija *rsort()* uređuje niz u opadajućem redosledu. Funkcija *ksort()*uređuje niz prema indeksima u rastućem, a funkcija *krsort()* u opadajućem redosledu.

***shuffle()***

Koristi se za proizvoljno raspoređivanje elemenata niza.

Primer:

|  |  |
| --- | --- |
| 1  2  3  4 | $my\_array = array("red","green","blue","yellow","purple");    shuffle($my\_array);  print\_r($my\_array); |

***array\_slice( $imeNiza, pozicija, dužina )***

U PHP-u je moguće kreirati novi niz kao podskup nekog već postojećeg pomoću funkcije *array\_slice()*. Na primer:

|  |  |
| --- | --- |
| 1  2 | $testarray = array("red", "green", "blue", "pink" );  $subArray = array\_slice( $testarray, 1, 2 ); |

Novi niz *$subArray* će sadržati dva elementa niza *$testArray* i to počevši od onog sa indeksom 1, tj.

[ 0 ] => green  
[ 1 ] => blue

***array\_merge( $imeNiza1,  $imeNiza2 )***

Ova funkcija vam omogućava da dva niza spojite u jedan. Na primer, sledeće naredbe:

|  |  |
| --- | --- |
| 1  2  3 | $array1 = array("red", "blue" );  $array2 = array("green", "yellow" );  $bigArray = array\_merge( $array1, $array2 ); |

će kreirati niz *$bigArray* sa sledećim elementima:

     $bigArray[ 0 ] = "red"  
     $bigArray[ 1 ] = "blue"  
     $bigArray[ 2 ] = "green"  
     $bigArray[ 3 ] = "yellow"

Prilikom korišćenja ove funkcije za spajanje dva asocijativna niza treba voditi računa o tome da u tim nizovima nema elemenata sa istim indeksom. Ukoliko dva elementa imaju isti string indeks, drugi od njih će biti prekopiran preko prvog.

***explode(),  implode()***

Ove dve funkcije se koriste za konvertovanje stringa u niz i obrnuto.

Funkcija *explode*() ima sledeću sintaksu

      $arrName = explode( "sign", $string );

Prvi argument ove funkcije (sign) određuje karakter koji će biti korišćen za razdvajanje stringa. Drugi argument je tekst koji se deli na delove. Svaki deo postaje jedan element niza.

Primer za explode():

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $someWords = "Please don't blow me to pieces.";    $wordChunks = explode(" ", $someWords);  for($i = 0; $i < count($wordChunks); $i++){      echo "Piece $i = $wordChunks[$i] <br />";  } |

Slično, moguće je elemente niza spojiti u string  pomoću funkcije *implode()*:

*$text = implode( "sign", $arrName );*

Primer za implode():

|  |  |
| --- | --- |
| 1  2  3  4 | $words = array("Please", "don't", "blow", "me", "to", "pieces.");    $str = implode(" ", $words);  echo $str; |

***array\_diff( $niz1, $niz2, ... )***

Funkcija identifikuje sve elemente koji se nalaze u prvom nizu ($niz1) ali se ne nalaze u ostalim nizovima.

Primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | $a1=array("a"=>"BMW","b"=>"Audi","c"=>"Fiat","d"=>"Opel");  $a2=array("e"=>"BMW","f"=>"Audi","g"=>"Peugeot");  $a3=array("a"=>"BMW","b"=>"Peugeot","h"=>"KIA");    $result=array\_diff($a1,$a2,$a3);  print\_r($result); |

Rezultat na strani je:

Array ( [c] => Fiat [d] => Opel )

***array\_sum()***

Ova funkcija sabira vrednosti koje su elementi niza.

Primer:

|  |  |
| --- | --- |
| 1  2 | $a=array(5,15,25);  echo array\_sum($a); |

Rezultat na strani je 45.

***array\_unique()***

Pomoću ove funkcije moguće je ukloniti iste elemente jednog niza.

Primer:

|  |  |
| --- | --- |
| 1  2 | $a=array("a"=>"BMW","b"=>"Audi","c"=>"BMW");  print\_r(array\_unique($a)); |

Rezultat na strani je:

Array ( [a] => BMW [b] => Audi )

**Koja od navedenih funkcije se koristi za sortiranje nizova?**

 substr()

 rsort()

 organise()

 pqmsort()

 srtm()

**Vežba 1**

***Problem:***

Definisan je sledeći niz:

|  |  |
| --- | --- |
| 1 | $arr = array( 2, 5, 1, 7, 4, 3, 8 ); |

Potrebno je napraviti aplikaciju koja će sortirati ovaj niz tako da vrednosti budu poređane od manje ka većoj.

Potrebno je prikazati sortirani niz na izlazu.

***Rešenje:***

Ovo rešenje podrazumeva selection sort algoritam, ali je moguće upotrebiti i različite druge algoritme sortiranja.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | <?php  $arr = array( 2, 5, 1, 7, 4, 3, 8 );  for( $i = 0; $i < sizeof( $arr ); $i++ )  {      $min = $i;      for( $u = $i; $u < sizeof( $arr ); $u++ )          {              if( $arr[ $u ] < $arr[ $min ] )                     $min = $u;          }          $tmp = $arr[ $i ];          $arr[ $i ] = $arr[ $min ];          $arr[ $min ] = $tmp;  }  print\_r( $arr );  ?> |

Sortirani niz na izlazu je:

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

Kreiramo promenljivu koja sadrži niz spreman za sortiranje i nazivamo je $arr. Upotrebom for petlje prolazimo kroz sve elemente niza, odnosno obezbeđujemo broj iteracija koji odgovara broju elemenata niza. Nakon završenog rada ove petlje niz će biti sortiran i spreman i možemo ga prikazati. Zato koristimo funkciju print\_r().  
  
Naš kod sada izgleda ovako:

|  |  |
| --- | --- |
| 1  2  3  4  5 | $arr = array( 2, 5, 1, 7, 4, 3, 8 );  for( $i = 0; $i < sizeof( $arr ); $i++ ){    }  print\_r( $arr ); |

U svakoj iteraciji očekujemo da je element sa indeksom $i najmanji, te zato postavljamo:

|  |  |
| --- | --- |
| 1 | $min = $i; |

Ipak, ovo je samo pretpostavka i ona mora biti proverena. Da bismo to uradili, element koji ima indeks $i, moramo uporediti sa ostalim elementima. Ovde nam može poslužiti nova unutrašnja for petlja.  
  
Sada naš kod izgleda ovako:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | $arr = array( 2, 5, 1, 7, 4, 3, 8 );  for( $i = 0; $i < sizeof( $arr ); $i++ )  {      $min = $i;      for( $u = $i; $u < sizeof( $arr ); $u++ ){      }  }  print\_r( $arr ); |

U svakoj iteraciji ove unutrašnje petlje vršimo proveru: $arr[ $u ] < $arr[ $min ] da bismo utvrdili odnos među testiranim elementima. Ukoliko je ovaj uslov ispunjen, potrebno je promeniti i vrednosti promenljive $min, čiji je cilj da sadrži indeks najmanjeg elementa u nizu.  
  
Kada petlja završi sve iteracije, ostaje da zamenimo mesta elementima u nizu. Najpre sa pozicije $i treba da preuzmemo element i da ga smestimo u neki privremeni prostor (jer ne želimo da ga izgubimo). To činimo na sledeći način:

|  |  |
| --- | --- |
| 1 | $tmp = $arr[ $i ]; |

Na mesto ovog elementa postavljamo onaj koji smo utvrdili kao najmanji:

|  |  |
| --- | --- |
| 1 | $arr[ $i ] = $arr[ $min ]; |

I na kraju, sa privremene pozicije vraćamo u niz (na upražnjeno mesto) element koji smo prethodno izvukli:

|  |  |
| --- | --- |
| 1 | $arr[ $min ] = $tmp; |

Ovim je naš zadatak završen.

**Vežba 2**

***Problem:***

Dat je niz:

|  |  |
| --- | --- |
| 1 | $arr = array( 0, 1, 2, 3, 4, 5, 6, 7, 8 ); |

Potrebno je napraviti kod koji će prikazati svakog člana niza pomnoženog sa prethodnim članom niza.

0  
0  
2  
6  
12  
20  
30  
42  
56

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  $arr = array(0, 1, 2, 3, 4, 5, 6, 7, 8);  for($i=0; $i<sizeof($arr); $i++){          if($i>0){              echo $arr[$i] \* $arr[$i-1]."<br>";          }else{              echo $arr[ $i ]."<br>";          }      }  ?> |

Ovim zadatkom prikazan je klasičan primer upotrebe nizova i petlji u kombinaciji. Suština ovog zadatka je zapravo samo u ideji za dobavljanje odgovarajućeg elementa iz niza i njegovog prethodnika. Ukoliko brojač $i koristimo kao indeks, onda je prethodni indeks svakako $i-1. Sada nam kao jedini problem ostaje situacija u kojoj bi brojač imao vrednost 0. Tada bismo po ovoj logici zatražili prethodni indeks, odnosno indeks -1, što bi izazvalo pojavljivanje greške. Kako se to ne bi dogodilo, postavljamo uslovljavanje:

|  |  |
| --- | --- |
| 1  2  3  4  5 | if($i>0){              echo $arr[$i] \* $arr[$i-1]."<br>";          }else{              echo $arr[ $i ]."<br>";          } |

**Vežba 3**

***Problem:***

Data je sledeća promenljiva:

*$lipsum = "Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.";*

Potrebno je napraviti program koji će prebrojati tačan broj svake reči u ovom stringu i emitovati rezultat na izlazu u sledećem formatu:

Lorem : 4  
Ipsum : 3  
is : 1  
simply : 1  
dummy : 2  
text : 2  
of : 4

***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 | <?php  $lipsum = "Lorem Ipsum is simply dummy text of the printing and  typesetting industry. Lorem Ipsum has been the industry's standard  dummy text ever since the 1500s, when an unknown printer took a  galley of type and scrambled it to make a type specimen book. It has  survived not only five centuries, but also the leap into electronic  typesetting, remaining essentially unchanged. It was popularised in  the 1960s with the release of Letraset sheets containing Lorem Ipsum  passages, and more recently with desktop publishing software like  Aldus PageMaker including versions of Lorem Ipsum.";    $arr = explode(" ", $lipsum);  $words = array();  $counts = array();  for($i=0; $i<sizeof($arr); $i++){          if(!in\_array($arr[$i], $words)){                  $words[] = $arr[ $i ];                  $counts[] = 1;              }else{                  for($u=0; $u<sizeof($words); $u++){                      if( $words[ $u ] == $arr[ $i ]){                          $counts[ $u ]++;                      }                  }              }      }  for($i=0; $i<sizeof($words); $i++){      echo($words[$i] . " : " . $counts[$i] . "<br>");  }  ?> |

Promenljiva $lipsum predstavlja string koji ćemo obrađivati. U promenljivu $arr smeštamo vrednost koju nam vrati funkcija explode(). Ova funkcija u našem slučaju prihvata dva argumenta. Prvi parametar je separator, odnosno karakter po kojem ćemo izvršiti razdvajanje stringa na elemente niza. Drugi argument je string od kojeg će biti kreiran niz $arr. Kako smo ovde postavili kao serparator razmak, tako će elementi našeg niza biti reči koje se nalaze u tekstu ($lipsum).  
  
Za dalji rad nam je potreban niz za reči i niz koji ćemo koristiti kao brojač. Zato kreiramo te nizove kao prazne:

|  |  |
| --- | --- |
| 1  2 | $words = array();  $counts = array(); |

Da bismo prošli kroz svaku reč, vršimo prolaz kroz sve elemente niza:

|  |  |
| --- | --- |
| 1 | for($i=0; $i<sizeof($arr); $i++){} |

Unutar ove petlje najpre treba proveriti da li se reč (element) ne nalazi u nizu. Ukoliko pronađena reč nije u nizu, treba je dodati i vrednost brojača za prvi slobodan index postaviti na 1. Sa druge strane, ukoliko se reč nalazi u nizu, izvršiće se alternativni scenario. Ovde se vrši onoliko iteracija koliko ima elemenata niza $words i za svaki se proverava uslov:

|  |  |
| --- | --- |
| 1 | $words[ $u ] == $arr[ $i ] |

Takođe, povećava se i vrednost elementa u nizu $counts koji odgovara brojaču unutrašnje petlje ($u).  
  
Na kraju, potrebno je samo prikazati rezultate oba niza:

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

**Vežba 4:**

***Problem:***

Potrebno je napisati program koji će pristupati elementima niza pomoću *foreach* petlje i sistema key=>value za niz mlečnih proizvoda (milk, chocolate, cheese) i njihove cene redosledno u vrednosti od 89, 75,105.

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  $price['milk'] = 89;  $price['chocolate'] = 75;  $price['cheese'] = 105;    foreach ($price as $key => $value) {      echo $key . '=>' . $value . '<br/>';  }    ?> |

Nakon izvršenja programa, na izlazu će biti sledeći ispis:

milk=>89  
chocolate=>75  
cheese=>105

Prikazana vežba predstavlja klasičan primer upotrebe foreach petlje. Upotreba foreach petlje prolazak kroz asocijativni niz čini jednostavnijim, jer se odmah mogu izdvojiti ključevi od vrednosti. U našem slučaju niz ima tri elementa i svaki od elemenata ima svoj ključ i svoju vrednost.