**Tipovi podataka**

**00:22:05**

Jedinica: 2 od 19

**+Rezime**

U ovoj lekciji naučićemo koji tipovi podataka postoje u PHP-u. Kada je reč o [promenljivama](https://www.link-elearning.com/linkdl/opisPojma.php?id=139094) i njihovim tipovima, PHP ima veoma fleksibilan sistem funkcionisanja. Štaviše, ovaj sistem je toliko fleksibilan da praktično ne moramo ni da znamo kojim tipovima rukujemo, niti koja su njihova ograničenja. PHP će, jednostavno, pokušati da izvuče ono što može iz informacija koje su mu date.

No, ovakva otvorenost prema programeru ne znači da tipovi uistinu i ne postoje, već da je rad sa njima [enkapsuliran](https://www.link-elearning.com/linkdl/opisPojma.php?id=139095) u ljušturu jednostavnosti, kako bi programiranje bilo lakše i brže.

**Tipovi u PHP-u**

U generalnoj klasifikaciji tipova podataka (i ovo važi za skoro sve programske jezike), razlikujemo [proste i složene tipove](https://www.link-elearning.com/linkdl/opisPojma.php?id=139096). Pored navedene, možemo koristiti i sledeću klasifikaciju za tipove podataka:

* + Skalarni tipovi,
  + Kompozitni tipovi,
  + Specijalni tipovi.

Prosti ili skalarni tipovi su oni tipovi koji podrazumevaju samo jednu vrednost u jednom trenutku i PHP poznaje četiri prosta tipa podataka:

* + **Integer:**Predstavlja celobrojni tip čija vrednost zavisi od magistrale procesora i okruženja (OS). Ukoliko procesor radi na 32-bita, veličina ovog tipa može biti od -2147483647 do 2147483647. Za 64-bitni procesor, maksimalna vrednost ovog tipa je do 9223372036854775807;
  + **Float:**Tip float koristi se u PHP-u za sve operacije sa pokretnim zarezom;
  + **String:**Tip string sadrži tekstualne vrednosti;
  + **Boolean:**Sadrži logičku vrednost (**true** ili **false**). Boolean predstavlja dva moguća stanja – tačno i netačno. Logičke vrednosti se uglavnom koriste za poređenje uslova. U PHP-u netačna vrednost odgovara jednoj od sledećih vrednosti: celobrojna 0, realna 0.0, prazan string, karakter '0' i konstanta NULL. Svaka druga vrednost se smatra za tačnu (true).

i dva složena (kompozitna) tipa.

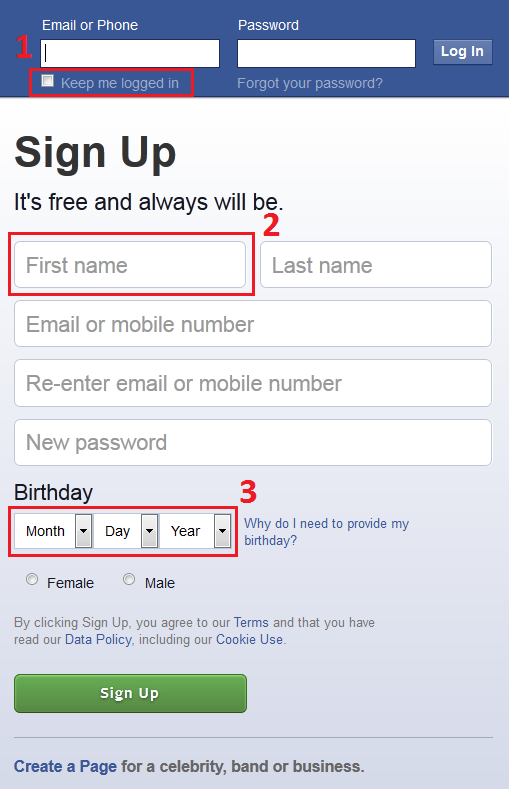
Složeni tipovi se obično tretiraju kao skladišta, odnosno kolekcije drugih podataka.

* + **Array:**Sadrži niz podataka. Ovaj niz sadrži podatke istih ili različitih tipova bilo koje vrste, pa čak i samih nizova. Ono što u sebi nosi promenljiva koja reprezentuje ovaj tip jeste zapravo memorijska adresa na kojoj kolekcija ovih podataka počinje;
  + **Object:**Sadrži objekat. Što se samih objekata tiče, oni su kompleksne strukture podataka koje sadrže podatke i funkcionalnost. Objekti su simbol objektno orijentisanog programiranja, i objektno orijentisanih programskih jezika, u koje spada i PHP, zbog čega ćemo objekte, kroz ovaj kurs, detaljno obraditi.

Pored ove podele, postoji i podela na numeričke tipove, stringove i bulove tipove. Ovo su, zapravo, podkategorije unutar generalne tipske kategorizacije.

Zbog čega nas ovo uopšte interesuje? Zašto bismo koristili nešto što u sebi sadrži tačnu ili netačnu vrednost, broj ili tekst?

Pogledajmo Facebook formu za logovanje ili prijavljivanje na sistem:



*2.1. Facebook Sign Up*

U mnoštvu informacija koje se u ovoj formi mogu postaviti, označene su tri. Prva (1) je Keep me logged in. Složićemo se da ova opcija može biti uključena ili isključena, te da nam za čuvanje ove informacije treba samo jedno od dva stanja. To se apsolutno poklapa sa boolean tipom opisanim u prethodnom delu lekcije. Kada podatak dođe do servera, pretvorićemo ga u ovaj tip i nastaviti da radimo sa njim dalje u tom obliku. Datum rođenja se očigledno unosi kroz tri padajuća menija (select kontrole). Svaki od njih sadrži listu brojeva, i to celih. Za sve tri informacije, idealan će biti upravo celobrojni tip (int/Integer), opisan ranije. Konačno, za podatke predstavljene kroz tekst boksove u formi, veoma verovatno će biti korišćen tip string.

**Numerički tipovi**

U ovu grupu tipova spadaju integer i float i za njih je karakteristično da čuvaju brojnu vrednost. PHP će, na osnovu načina na koji smo uneli vrednost, pretpostaviti i njen tip.

Već smo pomenuli tip integer. Ovaj tip podrazumeva označene vrednosti (negativne i pozitivne) u veličini koju dozvoljava sistem. Pritom, prihvata tri različita tipa notacije: decimalnu, oktalnu i heksadecimalnu.

Kada je u pitanju decimalna notacija, rukovanje je jednostavno, jer je to notacija kakvu poznajemo i koristimo u svakodnevnom životu. Na primer, da bismo predstavili broj deset, napisaćemo 10.

Oktalna i heksadecimalna notacija su nešto komplikovanije za upotrebu, ali se i ne koriste tako često. Zapravo, oktalna notacija najčešće se koristi prilikom rukovanja pravima u UNIX stilu (0666, 0667, 0777).

Obratite pažnju na oktalnu notaciju kada rukujete brojevima. Ova notacija aktivira se oznakom nula ispred broja, što greškom možete uraditi i prilikom rukovanja decimalnim brojem. Ovo može dovesti do nepravilnih rezultata prilikom izvršavanja programa.

Na primer, 100 u PHP-u označava broj 100, ali 0100 predstavlja broj 64.

Heksadecimalna notacija podrazumeva oznaku 0x (ili 0X, pošto ovaj vid notacije nije osetljiv na velika i mala slova), nakon koje se ispisuje heksadecimalna vrednost.

Float (naziva se još i double) takođe podrazumeva označene (pozitivne i negativne) vrednosti, ali, za razliku od integera, ovaj tip prihvata vrednosti sa pokretnim zarezom (decimalne vrednosti).

Float poznaje dva tipa notacije – decimalnu i eksponencijalnu (Decimal i Exponential).

Decimal je tradicionalni tip zapisa: 10.5, 0.55 itd., dok eksponencijalni ima malo ređe korišćenu formu. Eksponencijalni (normalizovani) zapis ćemo prepoznati po oznaci e (ili E - Takođe nije osetljivo na mala i velika slova), kojoj prethodi osnova broja, a sleduje broj koji označava broj nula iza množioca.

Na primer: 1e2 = 1 \* 100 = 100 ili 1.2e2 = 1.2 \* 100 = 120

Prilikom rukovanja sa numeričkim tipovima, treba obratiti pažnju na nekoliko sitnih, ali veoma važnih elemenata, koji mogu drastično uticati na tok programa.

* + Već smo pomenuli da veličina ovih vrednosti zavisi pre svega od sistema na kome se izvršavaju. To znači da prelaskom sa sistema na sistem naša aplikacija može dati drugačije rezultate, u zavisnosti od performansi sistema;
  + Float tipovi u PHP-u mogu dati neočekivane rezultate prilikom konverzije. Na primer, ukoliko konvertujemo sledeći izraz u tip integer: ((0.1 + 0.7) \* 10), rezultat te konverzacije će, iako bi po svoj logici trebalo da bude broj 8, biti broj 7. To je zato što je rezultat sabiranja 0.1 + 0.7 u PHP-u broj 7.999999. Prilikom konverzacije u integer, ovom broju se jednostavno skidaju decimale.

**Stringovi**

String je u PHP-u tip podataka koji sadrži sortiranu kolekciju binarnih vrednosti, koja može da sačinjava bilo kakvu binarnu strukturu (sliku, zvuk, tekst itd.). Ipak, kada se kaže string, misli se pre svega na tekstualni sadržaj, pa je najbolje da ga i zapamtimo kao nosioca tekstualnog sadržaja.

**Boolean**

O boolean tipu je dosta rečeno u samom nabrajanju tipova, pa ćemo pomenuti samo neke konverzacione osobenosti:

* + Prilikom konverzacije iz brojne u boolean vrednost, sve vrednosti koje nisu nula, biće konvertovane u true (čak i negativne vrednosti);
  + Prilikom konverzacije stringa u boolean vrednost, jedino stringovi koji nemaju sadržaj ili sadrže samo jedan karakter (0) biće konvertovani u false.

Pored pomenutih, PHP poznaje i dva specijalna tipa podataka.

**Resource**

Ovo je tip podataka koji je, u stvari, referenca na određeni izvor podataka. Ovo je poseban tip i postoje posebne funkcije koje ovim tipom rukuju. Fizički, ovaj tip predstavlja jedan tok podataka. Zato se, često, promenljive ovog tipa nazivaju i Resource Stream ili samo Stream.

Uzmimo za primer jedan tekstualni fajl. Taj fajl nije sastavni deo aplikacije, ali u njemu (na primer) postoje određeni podaci koji su potrebni aplikaciji da bi funkcionisala. Da bi programer pristupio ovom fajlu, potrebno je da ga na neki način tretira tokom izvršavanja programa. Ali, da bi to mogao da učini, potreban mu je objekat koji će reprezentovati taj fajl u kodu. Taj objekat naziva se resurs.

**NULL**

Ovo, zapravo, i nije tip podataka, već pre fiksna vrednost, koju obično imaju objekti kojima vrednost još uvek nije dodeljena ili je obrisana. Tip NULL može imati samo jednu vrednost – NULL.

**Konverzije tipova**

PHP se u većini slučajeva sam brine oko konverzije jednog tipa u drugi. Možete jednostavno napisati "1" + 2 i rezultat će biti 3, iako je jedan tip string, a drugi tip broj.

Ipak, ovakve implicitne konverzije funkcionišu kada su u pitanju tipovi prostih podataka, ali ne i u slučaju složenih tipova. Prost tip ne možete konvertovati u resurs. Sa druge strane, resurs hipotetički možete konvertovati u string, ali ćete pritom dobiti samo tekstualnu reprezentaciju tog resursa.

Da bismo konvertovali vrednost iz jednog tipa u drugi, koristimo male zagrade, na sledeći način:

**(tip u koji hoćemo da izvršimo konverziju)vrednost**

Na primer, ako bismo hteli da broj 10 pretvorimo u string, napisali bismo:

|  |  |
| --- | --- |
| 1 | (string)10; |

A ako bismo hteli da pretvorimo string u ceo broj, napisali bismo:

|  |  |
| --- | --- |
| 1 | (int)"10"; |

Naravno, da bi ovakva konverzija imala smisla, trebalo bi da pod navodnicima bude nešto što može da se uspešno prevede u broj (kao što je broj 10 u prethodnom primeru). Ali ako i ne obezbedimo ovakav podatak, php se neće pobuniti, već će izvršiti konverziju što bolje može. U sledećem primeru, kao rezultat konverzije dobićemo broj 0.

|  |  |
| --- | --- |
| 1 | (int)"a"; |

Sledeći primer konvertuje broj 015 u celobrojnu vrednost. Logično je da se praktično ništa ne dogodi, i da na izlazu dobijemo broj 15, ali rezultat neće biti takav.

|  |  |
| --- | --- |
| 1  2 | $x = 015;  echo (int)$x; |

Umesto broja 15, dobićemo broj 13. Ovo je u stvari potpuno normalno ponašanje jer je vrednost $x definisana kao oktalna (ima nulu na početku) i stoga, kada se pretvori u decimalnu nije 15, već 13. Zapravo, vrednost 13 bismo dobili čak i da ne izvršimo konverziju, osim ako ne odlučimo da formatiramo broj na drugačiji način na izlazu.

**Formatiranje brojeva**

Često će se pri ispisu brojeva javiti i potreba za njihovim formatiranjem, ukoliko želite da ispišete cifre razdvojene nekim znakom ili na primer da koristite separator za decimale. Za formatiranje u obliku celog broja koristi se [funkcija](https://www.link-elearning.com/linkdl/opisPojma.php?id=139088) **number\_format().**

|  |  |
| --- | --- |
| 1  2 | $number= 879.22;  echo number\_format($number); |

Ispis na izlazu biće: 879

Funkcija number\_format() zaokružuje broj koji formatira na najbliži ceo broj.

Vrednosti koje se postavljaju unutar zagrada pri pozivu funkcije nazivaju se argumenti ili parametri. Ove vrednosti funkcijama mogu davati dodatne informacije o tome kako treba da se ponašaju. Argumenti se odvajaju zarezima. Za formatiranje u obliku decimalnog broja, zadaje se broj decimalnih mesta kao drugi argument u pozivu funkcije. Ako kroz funkciju provučemo broj 879.223456:

|  |  |
| --- | --- |
| 1 | echo number\_format($number, 2); |

Ispis na izlazu biće: 879.22

Kao što uočavamo, funkcija number\_format() može funkcionisati samo sa jednim ili sa dva parametra. Takođe, ova funkcija može prihvatiti i treći i četvrti parametar koji je opcioni, poput drugog. Treći parametar predstavlja definiciju oznake koja će se koristiti za odvajanje decimalnog zapisa, dok četvrti parametar predstavlja oznaku za odvajanje hiljada u zapisu broja.

Primer:

|  |  |
| --- | --- |
| 1  2 | $x = 123456789.123;  echo number\_format($x, 2, "\*", "/"); |

Ispis na izlazu biće: 123/456/789\*12

Formatiranje novčanih iznosa biće takođe potrebno u radu, pa ćemo ukratko objasniti osnove formatiranja. Upotrebom funkcije **money\_format()** formatira se broj tako što se umeće znak za željenu valutu, separator decimalnih mesta i separator grupa od po tri cifre koje odgovaraju datom regionu. Kao prvi argument funkcije poziva se %n da bi se cifra ispisala u američkom formatu za valutu ili %i, koji predstavlja standardni format za valutu koja se upotrebljava međunarodno ili lokalno. Funkcija **setlocale()**podešava region i lokalne podatke.

Primer:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | <?php  $number = 2234.16;  setlocale(LC\_MONETARY, 'en\_US');  print money\_format('%n', $number);  print money\_format('%i',$number);    setlocale(LC\_MONETARY, 'fr\_FR');  print money\_format('%i', $number);    ?> |

|  |
| --- |
| **Napomena:**  Funkcija money\_format() će funkcionisati samo na Linux sistemu, što znači da na Windowsu nije podržana. |

Nakon izvršenja, na strani se pojavljuje sledeći rezultat:

SAD: $2,234.16  
USD: 2,234.16  
2 234.56 EUR

Formatiranje je moguće izvršiti i funkcijom printf. Ovo je nešto što osobe sa iskustvom u programskom jeziku C već znaju. U sledećem primeru definisani broj je prikazan na tri načina. Oktalno, decimalno i binarno, uz pomoć funkcije printf.

|  |  |
| --- | --- |
| 1  2 | $x = 015;  printf("Octal x is: %o, decimal is %d, and binary is %b",$x,$x,$x); |

O funkciji printf će biti još reči, pa ćemo samo ukratko reći da se u delu pod navodnicima nalazi tekst koji treba da bude prikazan, i da se u tom tekstu nalaze oznake na čijem mestu će se naći odgovarajući parametar iz liste parametara nakon dela pod navodnicima. Ove oznake počinju znakom % čemu sledi oznaka tipa; o je oktalni numerički tip, d je decimalni, a b binarni. U stringu imamo tri pojavljivanja, a nakon stringa tri parametra, od kojih svaki odgovara poziciji u stringu. Kao rezultat dobićemo sledeći ispis na strani:

Octal x is: 15, decimal is 13, and binary is 1101

**Datum i vreme**

Datum i vreme su veoma važni elementi u razvoju web strana i aplikacija. PHP omogućava da sa ovim podacima radite drugačije nego sa običnim stringovima. Kompjuter čuva datum i vreme u formatu koji se naziva timestamp. Ovaj format iskazuje određeni datum i vreme preko sekundi. S obzirom na to da je takav način prikazivanja vremena nerazumljiv za običnog korisnika, PHP ga konvertuje u neki od oblika koji ljudi koriste.

Neke od funkcija PHP-a za rad sa vremenom i datumom su:

* + *date(format)* – vraća trenutno vreme, formatirano prema instrukcijama u argumentu format;
  + *checkdate(mesec, dan, godina)* – proverava da li je datum validan, tj. da li su vrednosti navedene za mesec, dan i godinu u odgovarajućim rasponima;
  + *time()* – vraća trenutno vreme, iskazano u sekundama od 1. januara 1970.

U tabeli su navedeni validni formati koje možete da koristite sa funkcijom date():

|  |  |
| --- | --- |
| **a** | Štampa „am" ili „pm" |
| **A** | Štampa „AM" ili „PM" |
| **h** | Sati izraženi u rasponu od 01 do 12 |
| **H** | Sati izraženi u rasponu od 00 do 23 |
| **g** | Sati izraženi u rasponu od 1 do 12, bez vodećih nula |
| **G** | Sati izraženi u rasponu od 0 do 23, bez vodećih nula |
| **i** | Minuti (od 00 do 59) |
| **s** | Sekunde (od 00 do 59) |
| **d** | Dan u mesecu izražen sa dve cifre (01–31) |
| **D** | Dan u nedelji izražen skraćenim tekstom (Mon–Sun) |
| **l** | Dan u nedelji izražen tekstom (Monday–Sunday) |
| **F** | Pun naziv meseca (January–December) |
| **m** | Mesec izražen ciframa (1–12) |
| **Y** | Godina zapisana sa 4 cifre (2005) |
| **y** | Godina zapisana sa 2 cifre (05) |
| **s** | Engleski sufiksi (th, nd, st) |

*Tabela 2.1*

Kod koji sledi prikazuje trenutno vreme i datum u okviru web strane:

|  |  |
| --- | --- |
| 1  2  3  4  5 | <?php  echo 'Current date: '. date('d-n-Y');  echo '<br/>';  echo 'Current time:'. date('g:i:s a');  ?> |

Na izlazu nakon izvršenja ćemo dobiti:

*Current date*: 31-1-2013  
*Current time*: 11:33:26 am

Funkcija **getdate**() vraća niz parova ključ-vrednost. Vrednost za ključeve se dobavlja tako što se naziv ključa zapisuje pod navodnicima, unutar uglastih zagrada, odmah nakon naziva promenljive. Više o ovome ćete saznati u lekciji: Rad sa nizovima.

Povratne vrednosti funkcije getdate() su:

|  |  |
| --- | --- |
| **Value** | **Key** |
| Seconds | seconds |
| Minutes | minutes |
| Hours | hours |
| Day of the month | mday |
| Day of the week, numeric (Sunday is 0, Saturday is 6) | wday |
| Month, numeric | mon |
| Year, numeric (4 digits) | year |
| Day of the year, numeric (e.g., 299) | yday |
| Day of the week, textual, full (e.g., “Friday") | weekday |
| Month, textual, full (e.g., “January") | month |
| Seconds since epoch (what time( ) returns) | 0 |

*Tabela 2.2*

|  |  |
| --- | --- |
| 1  2  3  4 | <?php  $time= getdate();  printf('%s,%d,%d',$time['month'],$time['mday'],$time['year']);  ?> |

Da bismo kreirali novu promenljivu koja sadrži informacije o datumu i vremenu koje nije trenutno (dakle, nalazi se negde u prošlosti ili budućnosti), koristimo funkciju date\_create(). Ova funkcija prihata zapis željenog vremena.

Na pimer:

|  |  |
| --- | --- |
| 1  2 | $newDate = date\_create('2015-3-20 10:15:55');  echo date\_format($newDate, 'd.m.Y / H-i-s'); |

Nakon ovih linija koda, rezultat na strani će biti:

20.03.2015 / 10-15-55

**Klasa DateTime**

Bitno je napomenuti da korišćenje funkcija predviđenih za formatiranje i prikazivanje datuma i vremena nije jedini način za rad sa pomenutim podacima. Klasa DateTime u PHP-u je nešto naprednijih okvira od ovog kursa, ali važno je napomenuti mogućnosti olakšica prilikom pozivanja datuma i vremena. [Klasa](https://www.link-elearning.com/linkdl/opisPojma.php?id=139113) DateTime pripada objektno orijentisanom stilu i navešćemo razlike između dosadašnjeg ispisa i objektno orijentisanog. U prethodnim primerima za ispis na izlazu poput 2015-01-01 00:00:00 služili smo se sledećim kodom:

|  |  |
| --- | --- |
| 1  2  3  4 | <?php  $date = date\_create('2015-01-01');  echo date\_format($date, 'Y-m-d H:i:s');  ?> |

Prikaz objektno orijentisanog načina kodiranja:

|  |  |
| --- | --- |
| 1  2  3  4 | <?php  $date=new DateTime ('2015-01-01');  echo $date->format('Y-m-d H:i:s');  ?> |

S obzirom na to da pred sobom imamo oba načina kodiranja, i zahteva za ispis i formatiranje datuma i vremena, možete se odučiti za dalji rad.

**Označite kompozitni tip podatka.**

 Integer

 Boolean

 Array

 Float

 String

**Koju od navedenih vrednosti može imati Boolean tip podatka?**

 10.50

 my text

 TRUE

 array()

**Vežba 1**

***Problem:***

U aplikaciji identifikaciona šifra korisnika se pamti na sledeći način:

|  |  |
| --- | --- |
| 1  2  3 | <?php  $pass = 11793847250045684;  ?> |

Prilikom emitovanja ovog broja, na izlazu se pojavljuje sledeća vrednost:

1.1793847250046E+16

Nad ovim brojem (šifrom) ne vrše se nikakve aritmetičke operacije, ali je potrebno da na izlazu šifra bude adekvatno prikazana.

***Rešenje:***

Najbolje je velike brojne vrednosti koje ne zahtevaju aritmetičke operacije reprezentovati u vidu stringa:

|  |  |
| --- | --- |
| 1  2  3 | <?php  $x = "11793847250045684";  ?> |

**Vežba 2**

***Problem:***

U okviru PHP aplikacije nalazi se linija koja emituje podatke o tačnom vremenu:

|  |  |
| --- | --- |
| 1  2  3 | <?php  echo "Current time: " . Date("h:i") . "h, day is: " . Date("D/M/y");  ?> |

Linija emituje sledeći rezultat:

Current time: 11:45h, day is: Wed/Jan/13

Korisnik želi da prikazani datum bude u sledećem formatu:

Current time: 11:45, day is: 16/01/2013

***Rešenje:***

|  |  |
| --- | --- |
| 1  2  3 | <?php  echo "Current time: " . Date("h:i") . ", day is: " . Date("d/m/Y");  ?> |

U postavci zadatka je kao argument funkcije Date korišćen string sledeće strukture: "D/M/y". Ovim stringom definišemo format datuma koji će biti prikazan. Već u postavci zadatka se može uočiti da će ovakva definicija formata datuma vratiti sledeći zapis: Wed/Jan/13. U rešenju je string koji se prosleđuje funkciji izmenjen i izgleda ovako: "d/m/Y". Iz ovog razloga će biti promenjen i izlaz koji je prikazan na strani, i sada će datum biti zapisan ovako: 16/01/2013. U tabeli koja je data u sadržaju nastavne lekcije možete videti i ostale moguće varijante za formatiranje vremena i datuma.

**Vežba 3**

***Problem:***

U okviru aplikacije potrebno je napraviti sistem koji će pratiti da li je korisnik ulogovan ili ne. Potrebno je odrediti tip podatka u kome će biti skladištena ova informacija.

***Rešenje:***

Ovaj tip može biti boolean (true ili false), jer postoje samo dve moguće vrednosti, tačno ili netačno.