



UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

**Elektronske lekcije o osnovnim
pojmovima
programskog jezika PHP**

MASTER RAD

Student
Olivera Pavlović

Mentor
prof. dr Miroslav Marić

Beograd
2020.

Sadržaj

Uvod	3
1 Elektronske lekcije	4
2 Uvod u PHP	7
3 Sintaksa	8
3.1 Komentari	8
3.2 Velika i mala slova	8
3.3 Promenljive	9
4 Tipovi podataka	11
4.1 Celi brojevi	11
4.2 Brojevi u pokretnom zarezu	12
4.3 Logičke vrednosti	12
4.4 Objekti	12
4.5 Vrednost NULL	13
4.6 Resursi	14
5 Niske	14
5.1 Funkcije nad niskama	15
6 Nizovi	16
6.1 Sortiranje niza	18
7 Konstante	21
8 Operatori	22
8.1 Aritmetički operatori	22
8.2 Operatori dodele	23
8.3 Operatori poređenja	23
8.4 Operatori uvećanja i umanjenja	24
8.5 Logički operatori	25
8.6 Operatori nad niskama	26
8.7 Operatori nad nizovima	26
8.8 Bitovski operatori	27
8.9 Operator izbora	27
9 Grananja	28

10	Petlje	30
10.1	<i>While</i> petlja	30
10.2	<i>Do while</i> petlja	31
10.3	<i>For</i> petlja	32
10.4	<i>Foreach</i> petlja	33
11	Funkcije	34
11.1	Korisnički definisane funkcije	34
11.2	Argumenti funkcije i povratne vrednosti	35
12	Formulari	35
12.1	Metod GET	36
12.2	Metod POST	36
12.3	Polja za unos podataka	36
13	Validacija podataka	38
14	Aplikacija napravljena pomoću osnovnih elemenata programskog jezika PHP	41
15	Saveti za početnike	45
16	Napredne teme	46
	Zaključak	47
	Literatura	48

Uvod

U današnje vreme je nemoguće zamisliti život bez interneta i mogućnosti koje on pruža. Veb tehnologije se svakodnevno razvijaju i sve popularnije postaje učenje preko interneta. Jedna od platformi za ovakvo učenje je eŠkola veba. Nalazi se na adresi http://edusoft.matf.bg.ac.rs/eskola_veba/#/home. Sadrži besplatne kurseve više različitih veb tehnologija. Svaki kurs se sastoji iz lekcija na srpskom jeziku. Elektronske lekcije o osnovnim pojmovima programskog jezika PHP su namenjene svima koji se prvi put susreću sa ovim programskim jezikom.

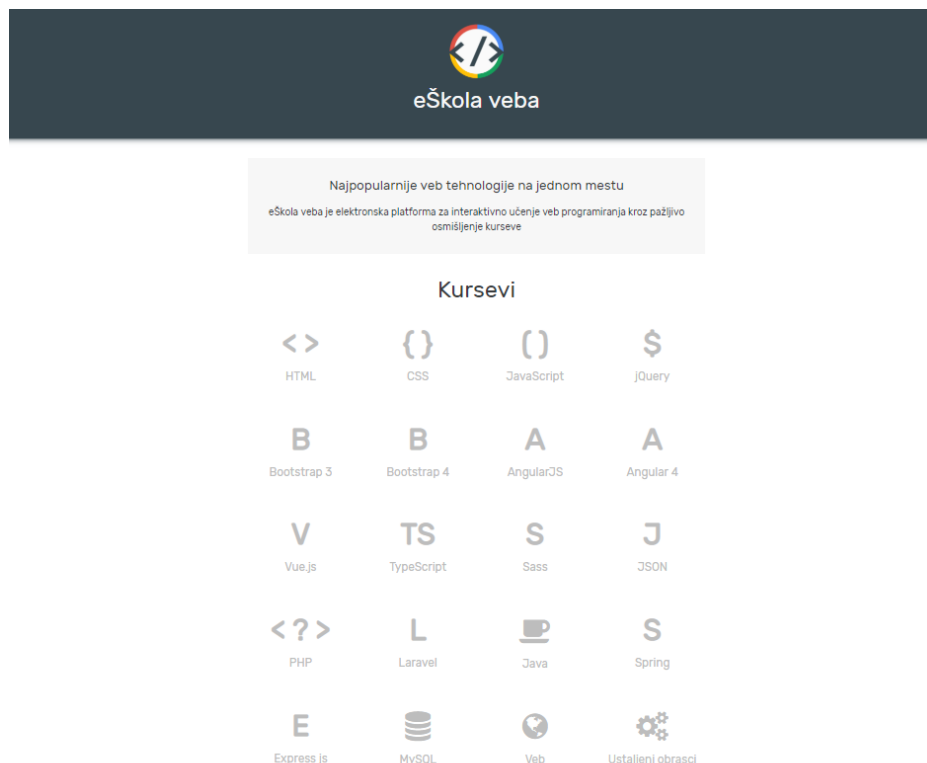
PHP je programski jezik za kreiranje dinamičkih veb stranica i serverskih komponenti veb aplikacija. Jednostavno se ugrađuje u HTML, što ga čini popularnim. PHP je slabo tipiziran programski jezik otvorenog koda. Sadrži ugrađenu biblioteku funkcija za rad sa različitim tipovima podataka i sadržaja kao što su niske, nizovi i slike. Posедуje funkcionalnosti koje omogućavaju objektno-orijentisano programiranje i jednostavan rad sa bazama podataka. Zbog svega navedenog PHP se često koristi u veb programiranju.

U ovom radu će biti prikazani osnovni pojmovi programskog jezika PHP. Na početku će biti opisano kako se pokreće PHP skripta na računaru i kako izgleda PHP kôd. Zatim će biti reči o sintaksi ovog programskog jezika i mogućnostima koje pruža korisnicima. Nakon toga će biti predstavljeni osnovni tipovi podataka koje PHP podržava kao što su celi brojevi, logičke vrednosti, ali i neke funkcije za rad sa niskama i nizovima. Programski jezik PHP podržava mnogo operacija i u narednim poglavljivama će biti opisani operatori i njihovo značenje. Većina programa se različito izvršava u različitim situacijama i tada se koriste petlje i grananja o čemu će takođe biti reči. U radu će biti prikazane funkcije koje su važan deo programiranja, kako ugrađene tako i one koje korisnik definiše. U poglavljima nakon toga biće prikazani formulari i njihovi elementi, kao i validacija podataka. Biće reči i o naprednim temama programskog jezika PHP za one koji savladaju ove lekcije.

Na kraju, biće prikazana veb aplikacija koja se može kreirati na osnovu obrađenih tema i saveti za početnike.

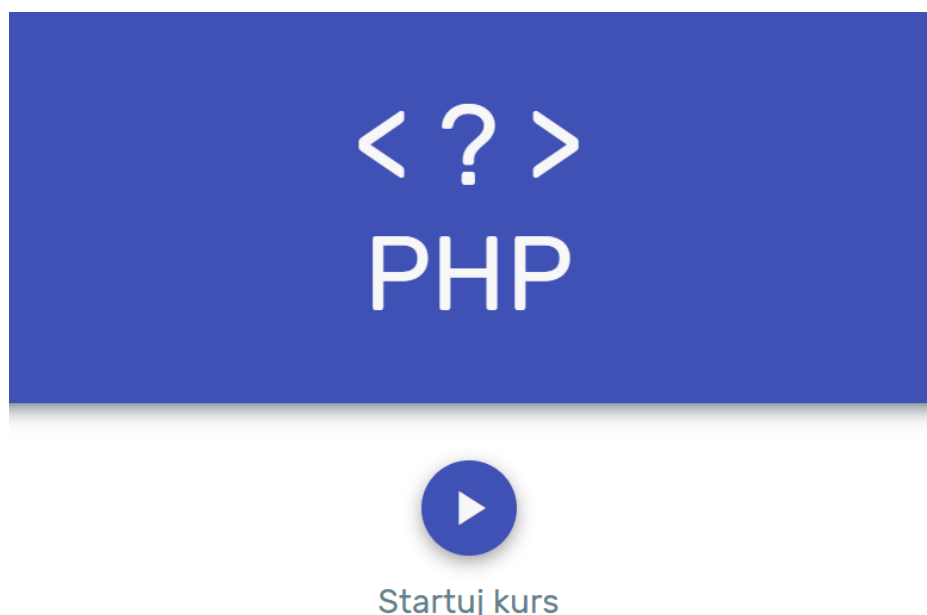
1 Elektronske lekcije

Elektronske lekcije o osnovnim pojmovima programskog jezika PHP su dostupne na platformi eŠkola veba. Lekcije su dostupne na linku http://edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/php i namenjene su svima koji žele da savladaju osnovne pojmove programskog jezika PHP. Sve lekcije su besplatne i dostupne na srpskom jeziku. Na slici 1 prikazan je izgled platforme.



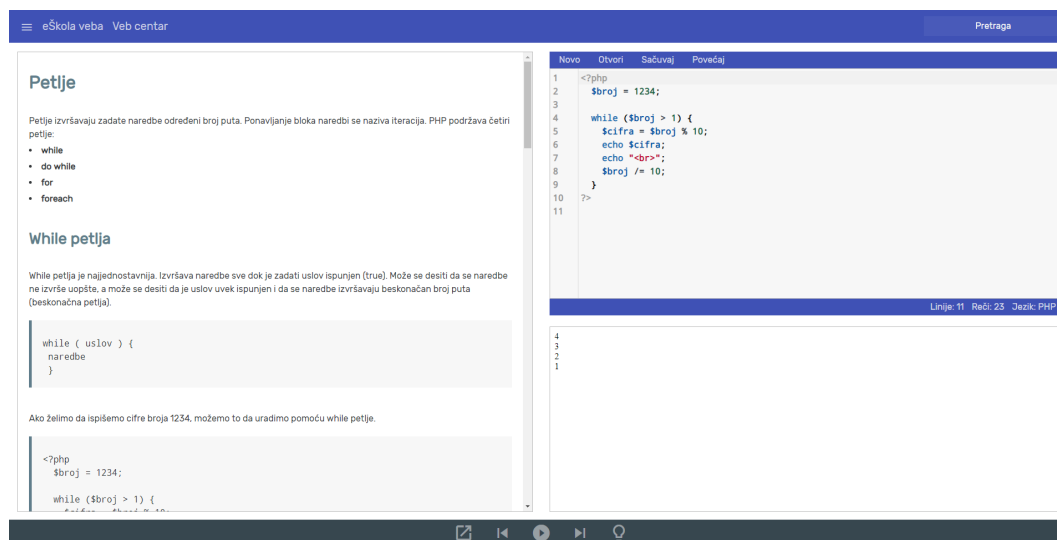
Slika 1: Dostupni kursevi na platformi eŠkola veba

Na platformi se nalaze kursevi o različitim veb tehnologijama. Svaki kurs se sastoji iz lekcija u kojima je sadržaj praćen primerima. Jedan od kurseva je i kurs o programskom jeziku PHP koji je kreiran za potrebe ovog rada. Kada se izabere ovaj kurs, otvara se početna strana kao na slici 2.



Slika 2: Naslovna strana kursa

Elektronske lekcije o osnovnim pojmovima programskog jezika PHP su kreirane da bi olakšale učenje osnovnih koncepata programskog jezika PHP. Svaka lekcija se sastoji iz tri dela. Prvi deo je sadržaj lekcije, drugi je primer, a treći je prikaz tog primera u veb pregledaču. Na slici 3 je prikazan izgled jedne lekcije.

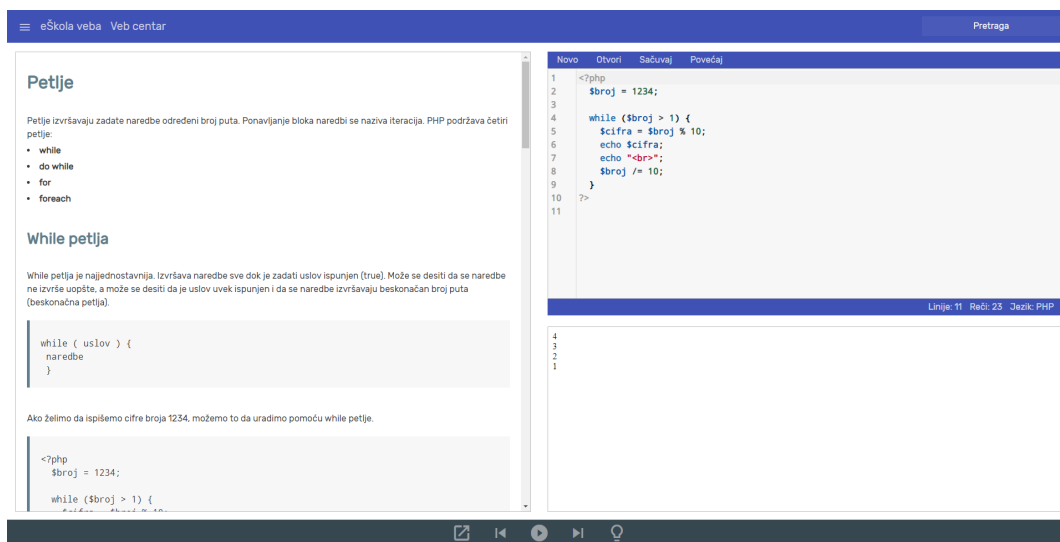


Slika 3: Izgled lekcije

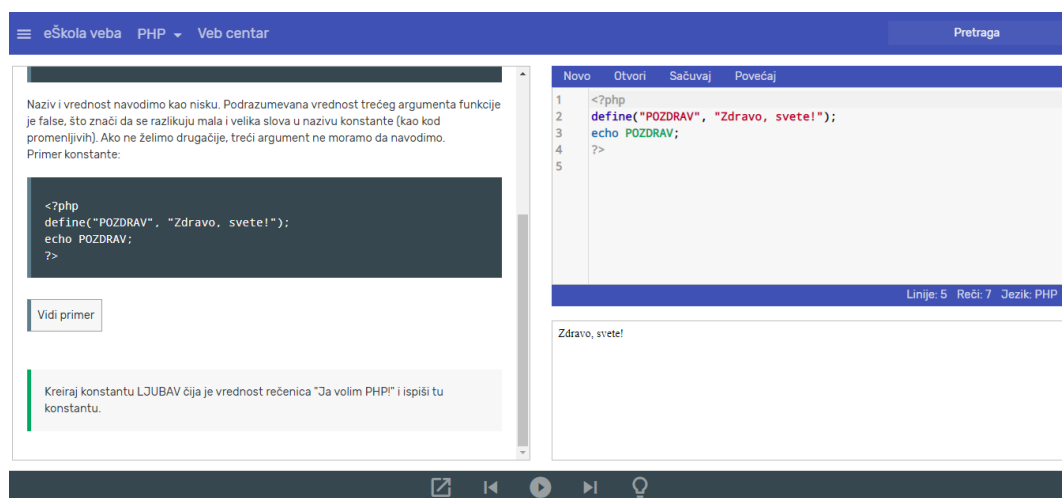
Za potrebe ovog rada kreirane su sledeće elektronske lekcije.

- Uvod u PHP;
- Sintaksa;
- Tipovi podataka;
- Niske;
- Nizovi;
- Konstante;
- Operatori;
- Grananja;
- Petlje;
- Funkcije;
- Formulari;
- Validacija podataka.

U sadržaju su pomoću definicija i primera objašnjeni pojmovi vezani za konkretnu lekciju. Primeri se mogu direktno pokrenuti sa platforme i menjati. Na kraju svake lekcije se nalazi zadatak koji treba da testira razumevanje pojmova objašnjenih u lekciji. Svaki zadatak ima priloženo rešenje. Na slici 4 je prikazan jedan od primera iz lekcije, a na slici 5 je prikazan jedan od zadataka.



Slika 4: Prikaz jednog primera



Slika 5: Prikaz jednog zadatka

Lekcije su kreirane sistematično, prvo su objašnjeni osnovni pojmovi, zatim složeniji. Stari pojmovi se koriste pri uvođenju novih. Cilj ovakvog pristupa je obnavljanje naučenog gradiva i otklanjanje nejasnoća iz prethodnog gradiva.

2 Uvod u PHP

PHP (PHP: Hypertext Preprocessor) je popularan skriptni jezik otvorenog koda. Namenjen je za izradu dinamičkih stranica koje mogu da komuniciraju sa bazama podataka i izvodi se na strani servera. Veoma je jednostavan i tolerantan jezik. Potrebno je u osnovi poznavati HTML, CSS i JavaScript da bi se lakše naučio PHP.

PHP nije strogo tipiziran, što znači da ista promenljiva može menjati tip podatka koji sadrži (iako se to ne preporučuje). Promenljive je moguće napraviti kad god je to potrebno.

Da bi se pokrenuo PHP, potreban je lokalni veb server i tekst editor. Postoji više besplatnih veb servera kao što su EasyPHP, XAMPP i WampServer za Windows i LampServer za Linux. Pogodni tekst editoru su Notepad++ i Atom, ali ima i drugih. PHP fajlovi imaju ekstenziju *.php* i mogu da sadrže tekst, HTML, CSS, JavaScript i PHP kôd.

PHP kôd se može nalaziti bilo gde u dokumentu. Počinje sa *<?php* i završava se sa *? >*. Svaka linija koda između ove dve komande završava se sa *(;)*.

Za ispisivanje teksta i vrednosti promenljive koriste se naredbe *echo* i *print*. *Echo* nema povratnu vrednost, a *print* uvek vraća vrednost i zbog toga se može tretirati kao funkcija i koristiti u izrazima.

Primer 1: Poziv funkcije *echo*

```
1 <?php
2     echo "Zdravo, svete!";
3 ?>
```

Rezultat primera 1 je:

```
1 Zdravo, svete!
```

Funkcija *var_dump()* ispisuje vrednost i tip promenljive.

Primer 2: Poziv funkcije *var_dump()*

```
1 <?php
2     $a = 11;
3     var_dump($a);
4 ?>
```

Rezultat primera 2 je:

```
1 int(11)
```

jer je 11 ceo broj (*int*).

3 Sintaksa

3.1 Komentari

Komentari su pomoćni opisi koda koji služe za lakše razumevanje koda. Prilikom pisanja programa trebalo bi napisati komentare koji objašnjavaju šta koji deo koda radi, jer se programi takođe i čitaju.

Komentari se mogu pisati na više načina. Jednolinijski komentar se piše posle karaktera *//*. To je kratak komentar koji može da se doda i na kraju linije.

```
// Ovo je jednolinijski komentar
$a="Zdravo"; //I ovo je jednolinijski komentar
```

Višelinijski (blok) komentari se pišu između */* */*.

```
/* Ovo je
viselinijski komentar */
```

3.2 Velika i mala slova

Programski jezik PHP razlikuje velika i mala slova. Ključne reči (*if*, *else*, *while* i druge), klase, funkcije i funkcije koje korisnik definiše mogu se pisati i malim i velikim slovima, ali ne i promenljive.

U sledećem primeru će biti prikazan poziv funkcije *echo* na tri načina koja se razlikuju po upotrebi velikih i malih slova.

Primer 3: Poziv funkcije *echo* na tri načina

```
1 <?php
2     ECHO "Zdravo, svete!<br>";
3     echo "Zdravo, svete!<br>";
4     EcHo "Zdravo, svete!<br>";
5 ?>
```

Ove tri *echo* naredbe su ekvivalentne i ispravne i imaju isti rezultat:

```
1     Zdravo, svete!
```

3.3 Promenljive

Promenljive služe za čuvanje informacija. Ispred naziva promenljive stoji znak \$. Naziv može da počinje slovom ili donjom crtom, ali ne i brojem. Može da sadrži mala i velika slova, brojeve i donju crtu. U nazivima promenljivih se razlikuju mala i velika slova. Na primer, *\$ime* i *\$Ime* su dve različite promenljive.

Primer 4: Primeri promenljivih i upotreba malih i velikih slova u nazivu promenljivih

```
1 <?php
2     $pozdrav = "Zdravo, svete!";
3     $x = 5;
4     $y = 10.5;
5     $ime = "Pera";
6     $Ime = "Mika";
7
8     echo $pozdrav;
9     echo "<br>";
10    echo $x;
11    echo "<br>";
12    echo $y;
13    echo "<br>";
14    echo $ime;
15    echo "<br>";
16    echo $Ime;
17 ?>
```

Rezultat primera 4 je:

```
1     Zdravo, svete!
2     5
3     10.5
4     Pera
5     Mika
```

Promenljive se ne deklariraju ni na kakav poseban način. Postaju definisane kada im se prvi put dodeli vrednost. Ako promenljivoj treba dodeliti tekst, potrebno je navesti ga između navodnika kao što je urađeno sa promenljivom *\$pozdrav* u prethodnom primeru.

Naredba *echo* se često koristi za ispis podataka na ekran.

Primer 5: Ispisivanje vrednosti promenljive pomoću funkcije *echo*

```
1 <?php
2     $p = "PHP";
3     echo "Ja volim $p!";
4 ?>
```

Rezultat primera 5 je:

```
1     Ja volim PHP!
```

Naredni primer ispisuje zbir dve promenljive, odnosno 11.

Primer 6: Zbir dve promenljive

```
1 <?php
2     $x = 8;
3     $y = 3;
4
5     echo $x + $y;
6 ?>
```

U prethodnim primerima nije bilo neophodno navesti tip promenljive. PHP automatski pretvara promenljivu u odgovarajući tip na osnovu njene vrednosti.

Već je rečeno da je promenljive moguće definisati kada god je to potrebno. Oblast važenja promenljivih je kontekst u kome su definisane. Postoje tri vrste promenljivih i to su *lokalne*, *globalne* i *statičke*. Promenljive koje su definisane na globalnom nivou ne važe unutar funkcija i obrnuto. Da bi se koristila *globalna* promenljiva u funkciji, potrebno je koristiti reč *global* ispred naziva promenljive. U suprotnom, nije moguće promeniti globalnu promenljivu u funkciji kao što je prikazano u primeru koji sledi.

Primer 7: Upotreba *Globalna* promenljiva i funkcija

```
1 <?php
2     $x = 5; // globalna promenljiva
3
4     function f() {
5         $x = 10; // lokalna promenljiva
6         echo "Vrednost promenljive x unutar funkcije je $x.";
7     }
8
9     f(); //poziv funkcije f
10    //Kasnije ce biti vise reci o funkcijama
11
12    echo "<br>";
13    echo "Vrednost promenljive x izvan funkcije je $x.";
14 ?>
```

Rezultat primera 7 je:

```
1     Vrednost promenljive x unutar funkcije je 10.
2     Vrednost promenljive x izvan funkcije je 5.
```

Nakon reči *global* se može navesti više promenljivih. To je još jedan primer upotrebe *globalne* promenljive u funkciji i prikazan je u sledećem primeru čiji je rezultat 15.

Primer 8: Drugi način upotrebe *globalne* promenljive u funkciji

```
1 <?php
2     $x = 5;
3     $y = 10;
4
5     function f() {
6         global $x, $y;
7         $y = $x + $y;
8         echo $y;
9     }
10
11     f();
12 ?>
```

Rezultat primera 8 je 15.

Unutar funkcije je moguće definisati i *statičku* promenljivu, koja jeste *lokalna*, ali ne gubi vrednost kada se izvršavanje funkcije završi. *Statičke* promenljive mogu se deklarirati upotrebom samo konkretne, konstantne vrednosti, a ne i izraza.

4 Tipovi podataka

Tip podatka predstavlja skup vrednosti i operacije koje se mogu obaviti nad tim skupom. U programiranju se koriste različiti podaci kao što su brojevi, tekstualni podaci, nizovi i drugi. S obzirom da se svi podaci u memoriji predstavljaju kao niz bitova, tip podatka zapravo daje značenje tom nizu.

Programski jezik PHP razlikuje više tipova podataka. Osnovni tipovi su:

- niske,
- celi brojevi,
- brojevi u pokretnom zarezu (brojevi sa decimalama, racionalni brojevi),
- logičke vrednosti,
- nizovi,
- objekti,
- vrednost NULL,
- resursi.

4.1 Celi brojevi

Celi brojevi (integer) su brojevi između -2147483648 i 2147483647 . Mogu se zapisati u dekadnom, oktalnom, heksadekadnom i binarnom sistemu. U kom sistemu će broj biti zapisan, zavisi od prefiksa:

- 0 (nula) - oktalni sistem,
- 0x (nula-iks) - heksadekadni sistem,
- 0b (nula-b) - binarni sistem.

Primer 9: Zapisi brojeva u različitim sistemima

```

1 <?php
2     $a = 1234;    //pozitivan broj
3     $b = -258;    //negativan broj
4     $c = 0253;    //oktalni broj
5     $d = 0x2F;    //heksadekadni broj
6     $e = 0b101;   //binarni broj
7 ?>

```

4.2 Brojevi u pokretnom zarezu

Brojevi u pokretnom zarezu (float, double ili realni brojevi) su realni brojevi sa decimalama. Decimalni znak je uvek tačka.

Primer 10: Ispisivanje broja u pokretnom zarezu

```

1 <?php
2     $a = 1.234;
3     echo $a;
4 ?>

```

Rezultat primera 10 je 1.234.

4.3 Logičke vrednosti

Postoje dve logičke vrednosti (boolean) i to su *true* i *false*. Najčešće se koriste u ispitivanju uslova i mogu da budu rezultat logičkih izraza i funkcija.

4.4 Objekti

PHP podržava i objektno-orijentisano programiranje. Objekat, koji se eksplicitno deklarise, čuva podatke i informacije o njihovoj obradi. Da bi se koristili objekti, prvo je potrebno napraviti klasu. To se radi pomoću ključne reči *class*. Klasa je struktura koja sadrži svojstva i metode što ilustruje sledeći kôd.

U sledećem primeru je prikazana klasa `Pas` koja ima svojstvo `$pozdrav` i metod `javi_se`. Konkretni objekat te klase je `$Bobi` koji ima `Zdravo! Ja sam Bobi.` za vrednost svojstva `$pozdrav`.

Primer 11: Primer klase

```
1 <?php
2     class Pas{
3
4         var $pozdrav;
5
6         function javi_se(){
7             echo $this->pozdrav;
8         }
9
10    }
11
12    //kreiranje instance objekta
13    $Bobi=new Pas();
14
15    $Bobi->pozdrav="Zdravo! Ja sam Bobi.";
16
17    //pozivanje funkcije javi_se
18    $Bobi->javi_se();
19 ?>
```

Rezultat primera 11 je:

```
1 Zdravo! Ja sam Bobi.
```

Napomena 1 Upotreba `this->` pre imena promenljive omogućava pristup istoj i može se koristiti za ispis ili menjanje podataka. Na isti način se pristupa funkcijama.

4.5 Vrednost NULL

NULL je poseban tip podatka i može da ima samo jednu vrednost, a to je NULL. Promenljiva tipa NULL nema dodeljenu vrednost.

Napomena 2 Ako se kreiranoj promenljivoj ne dodeli vrednost, njena vrednost postaje NULL.

Vrednost promenljive se briše tako što joj se dodeli vrednost NULL.

Primer 12: Brisanje vrednosti promenljive

```
1 <?php
2     $x = "Zdravo, svete!";
3     $x = null;
4
5     echo $x; //ne ispisuje nista
6     var_dump($x);
7 ?>
```

Rezultat primera 12 je NULL.

4.6 Resursi

Resursi (resources), koji predstavljaju napredni PHP, su poseban tip podataka i predstavljaju referencu ka funkciji ili resursu koji je izvan PHP skripte. Primer resursa je konekcija sa bazom podataka, jer se baza nalazi izvan PHP skripte.

Primer 13: Konekcija sa bazom

```
1 <?php
2 function konektuj(){
3     global $veza;
4     $veza = mysqli_connect('localhost','root','','kviz');
5     mysqli_set_charset($veza,"utf8");
6     if(mysqli_connect_errno($veza))
7         die("Greka prilikom povezivanja");
8 }
9
10 function diskonektuj(){
11     global $veza;
12     mysqli_close($veza);
13 }
14 ?>
```

5 Niske

Niske (string) su nizovi karaktera koji se navode između jednostrukih ili dvostrukih navodnika. Sadržaj se ispisuje različito zavisno od vrste navodnika koja se koristi. Sadržaj promenljive će se ispisati kada se upotrebe dupli navodnici, a ispisaće se naziv promenljive sa znakom \$ kada se upotrebe jednostruki. Sledeća dva primera prikazuju ovu razliku.

Primer 14: Ispisivanje sa duplim navodnicima

```
1 <?php
2 $ime="PHP";
3 echo "Moje ime je $ime.";
4 ?>
```

Rezultat primera 14 je:

```
1 Moje ime je PHP.
```

Primer 15: Ispisivanje sa jednostrukim navodnicima

```
1 <?php
2 $ime="PHP";
3 echo 'Moje ime je $ime.';
4 ?>
```

Rezultat primera 15 je:

```
1 Moje ime je $ime.
```

Dvostruki navodnici omogućavaju korišćenje specijalnih znakova koji se navode iza obrnute kose crte (\) i koriste se za preskakanje određenog znaka unutar niske.

Pomoću tačke (.) se spajaju stringovi.

Primer 16: Spajanje stringova

```
1 <?php
2     $niska1 = "Ja";
3     $niska2 = "ucim";
4
5     echo $niska1 . " " . $niska2 . " " . "veb programiranje.";
6 ?>
```

Rezultat primera 16 je:

```
1 Ja ucim veb programiranje.
```

5.1 Funkcije nad niskama

PHP podržava više funkcija za rad sa niskama. U nastavku će biti navedeno par funkcija koje se često koriste.

Funkcija *strlen()* vraća dužinu niske (stringa), a to je 14 u sledećem primeru.

Primer 17: Poziv funkcije *strlen()*

```
1 <?php
2     echo strlen("Zdravo, svete!");
3 ?>
```

Funkcija *str_word_count()* vraća broj reči koje niska sadrži.

Primer 18: Poziv funkcije *str_word_count()*

```
1 <?php
2     echo str_word_count("Ja volim veb programiranje.");
3 ?>
```

Rezultat primera 18 je 4.

Funkcija *strrev()* obrće nisku.

Primer 19: Poziv funkcije *strrev()*

```
1 <?php
2     echo strrev("Zdravo, svete!");
3 ?>
```

Rezultat primera 19 je:

```
1 !etevs ,ovardZ
```

Funkcija *strpos()* traži zadati tekst u stringu i vraća poziciju prvog poklapanja. U suprotnom vraća *false*.

Primer 20: Poziv funkcije *strpos()*

```
1 <?php
2     echo strpos("Zdravo, svete!", "svete");
3 ?>
```

Kako se reč **svete** nalazi u stringu **Zdravo, svete!**, rezultat primera 20 je 8.

Funkcija *str_replace()* menja određene znakove niske sa drugim znakovima. U primeru 21 se reč **svete** zamenjuje sa **Marija**.

Primer 21: Poziv funkcije *str_replace()*

```
1 <?php
2     echo str_replace("svete", "Marija", "Zdravo, svete!");
3 ?>
```

Rezultat primera 21 je:

```
1     Zdravo, Marija!
```

6 Nizovi

Niz (array) je složena promenljiva koja čuva više vrednosti. Vrednosti mogu biti istog ili različitog tipa, a mogu biti i drugi nizovi. Svaku vrednost (član) niza jedinstveno određuje jedan ili više indeksa. Članovima se može pristupiti preko naziva niza i indeksa.

Platforma eŠkola veba ima više kurseva i potrebno je napraviti promenljivu za svaki kurs.

Primer 22: Promenljive koje predstavljaju kurseve

```
1 <?php
2     $kurs1 = "HTML";
3     $kurs2 = "CSS";
4     $kurs3 = "PHP";
5     $kurs4 = "Laravel";
6     $kurs5 = "Spring"; //i tako dalje
7 ?>
```

S obzirom da ima puno kurseva, biće i mnogo promenljivih. Kada bi bilo 300 kurseva ne bi bilo lako pronaći određeni kurs. Bolje rešenje je da se napravi niz. Niz se kreira pomoću funkcije *array()*.

Nizovi mogu biti:

- numerisani - pamte vrednosti pod rednim brojem počev od nule,
- asocijativni - pamte podatke po sistemu ključ-vrednost,
- višedimenzioni - spadaju u napredni PHP.

Kursevi eŠkole veba se mogu smestiti u numerisani niz.

Primer 23: Numerisani niz

```
1 <?php
2 $kursevi = array("HTML", "CSS", "PHP", "Laravel", "Spring");
3 ?>
```

U ovom slučaju su indeksi automatski dodeljeni, a moguće ih je dodeliti i ručno.

Primer 24: Ručno dodeljivanje indeksa

```
1 <?php
2 $kursevi[0] = "HTML";
3 $kursevi[1] = "CSS";
4 $kursevi[2] = "PHP";
5 $kursevi[3] = "Laravel";
6 $kursevi[4] = "Spring";
7 ?>
```

Elementu niza se lako pristupa preko njegovog indeksa. Nakon naziva niza se navodi indeks između uglastih zagrada.

Primer 25: Pristupanje elementu niza preko indeksa

```
1 <?php
2 $kursevi = array("HTML", "CSS", "PHP", "Laravel", "Spring");
3 echo "Ja volim " . $kursevi[2] . ".";
4 ?>
```

Rezultat primera 25 je:

```
1 Ja volim PHP.
```

Funkcija `count()` vraća dužinu (broj elemenata) niza.

Primer 26: Poziv funkcije `count()`

```
1 <?php
2 $kursevi = array("HTML", "CSS", "PHP", "Laravel", "Spring");
3 echo "Duzina niza je " . count($kursevi) . ".";
4 ?>
```

Rezultat primera 26 je:

```
1 Duzina niza je 5.
```

Asocijativni niz se može kreirati na dva načina prikazana u sledećim primerima. Prvi način je pomoću funkcije `array`, a drugi način je da se svakom ključu dodeli njegova vrednost.

Primer 27: Kreiranje asocijativnog niza pomoću funkcije `array`

```
1 <?php
2 $kosarkasi = array("Teodosic"=>"1", "Bogdanovic"=>"2", "Jokic"=>"3");
3 ?>
```

Primer 28: Kreiranje asocijativnog niza bez funkcije *array*

```
1 <?php
2 $kosarkasi["Teodosic"] = "1";
3 $kosarkasi["Bogdanovic"] = "2";
4 $kosarkasi["Jokic"] = "3";
5 ?>
```

Ključ može da bude ceo broj ili string, a vrednost može biti bilo kog tipa. Vrednosti pod određenim ključem se može pristupiti tako što se nakon naziva niza navede ključ između uglastih zagrada. Može se reći da se ključ posmatra kao indeks.

Primer 29: Pristup vrednosti funkcije preko ključa

```
1 <?php
2 $kosarkasi = array("Teodosic"=>"1", "Bogdanovic"=>"2", "Jokic"=>"3");
3 echo "Teodosic nosi dres sa brojem " . $kosarkasi["Teodosic"] . ".";
4 ?>
```

Rezultat primera 29 je:

```
1 Teodosic nosi dres sa brojem 1.
```

6.1 Sortiranje niza

Funkcije za sortiranje niza su:

- `sort()` - sortira niz u rastućem poretku,
- `rsort()` - sortira niz u opadajućem poretku,
- `asort()` - sortira asocijativni niz u rastućem poretku u odnosu na vrednost,
- `ksort()` - sortira asocijativni niz u rastućem poretku u odnosu na ključ,
- `arsort()` - sortira asocijativni niz u opadajućem poretku u odnosu na vrednost,
- `krsort()` - sortira asocijativni niz u opadajućem poretku u odnosu na ključ.

Napomena 3 Za prolazak kroz nizove i ispis njihovih elemenata se koristi *for* i *foreach* petlja o kojima će više reči biti u nastavku.

U primeru 30 je prikazan poziv funkcije *sort()* i prolazak petljom kroz sortiran niz kurseva u rastućem poretku (po alfabetu).

Primer 30: Poziv funkcije *sort()*

```
1 <?php
2 $kursevi = array("HTML", "CSS", "PHP", "Laravel", "Spring");
3 sort($kursevi);
4 $duzina_niza = count($kursevi);
5
6 for ($i = 0; $i < $duzina_niza; $i++) {
7     echo $kursevi[$i];
8     echo "<br>";
9 }
10 ?>
```

Rezultat primera 30 je:

```
1 CSS
2 HTML
3 Laravel
4 PHP
5 Spring
```

Primer 31 prikazuje poziv funkcije *rsort()* i prolazak petljom kroz sortiran niz kurseva u opadajućem poretku (od većeg broja ka manjem).

Primer 31: Poziv funkcije *rsort()*

```
1 <?php
2 $brojevi = array(2, 11, 8, 3, 13);
3 rsort($brojevi);
4 $duzina_niza = count($brojevi);
5
6 for ($i = 0; $i < $duzina_niza; $i++) {
7     echo $brojevi[$i];
8     echo "<br>";
9 }
10 ?>
```

Rezultat primera 31 je:

```
1 13
2 11
3 8
4 3
5 2
```

Funkcija *asort()* sortira asocijativni niz u rastućem poretku u odnosu na vrednost i to je prikazano u primeru 32.

Primer 32: Poziv funkcije *asort()*

```
1 <?php
2 $kosarkasi = array("Teodosic"=>"1", "Bogdanovic"=>"2", "Jokic"=>"3");
3 asort($kosarkasi);
4
5 foreach ($kosarkasi as $kljuc => $vrednost) {
6     echo "Kljuc = " . $kljuc . ", Vrednost = " . $vrednost;
7     echo "<br>";
8 }
9 ?>
```

Rezultat primera 32 je:

```
1 Kljuc = Teodosic, Vrednost = 1
2 Kljuc = Bogdanovic, Vrednost = 2
3 Kljuc = Jokic, Vrednost = 3
```

Asocijativni niz se sortira u rastućem poretку u odnosu na ključ pomoću funkcije *ksort()*.

Primer 33: Poziv funkcije *ksort()*

```
1 <?php
2 $kosarkasi = array("Teodosic"=>"1", "Bogdanovic"=>"2", "Jokic"=>"3");
3 ksort($kosarkasi);
4
5 foreach ($kosarkasi as $kljuc => $vrednost) {
6     echo "Kljuc = " . $kljuc . ", Vrednost = " . $vrednost;
7     echo "<br>";
8 }
9 ?>
```

Rezultat primera 33 je:

```
1 Kljuc = Bogdanovic, Vrednost = 2
2 Kljuc = Jokic, Vrednost = 3
3 Kljuc = Teodosic, Vrednost = 1
```

Da bi se asocijativni niz sortirao u opadajućem poretку u odnosu na vrednost koristi se funkcija *arsort()*.

Primer 34: Poziv funkcije *arsort()*

```
1 <?php
2 $kosarkasi = array("Teodosic"=>"1", "Bogdanovic"=>"2", "Jokic"=>"3");
3 arsort($kosarkasi);
4
5 foreach ($kosarkasi as $kljuc => $vrednost) {
6     echo "Kljuc = " . $kljuc . ", Vrednost = " . $vrednost;
7     echo "<br>";
8 }
9 ?>
```

Rezultat primera 34 je:

```
1 Kljuc = Jokic, Vrednost = 3
2 Kljuc = Bogdanovic, Vrednost = 2
3 Kljuc = Teodosic, Vrednost = 1
```

U primeru 35 je prikazano kako se asocijativni niz sortira u opadajućem poretку u odnosu na ključ pomoću funkcije *krsort()*.

Primer 35: Poziv funkcije *krsort()*

```
1 <?php
2 $kosarkasi = array("Teodosic"=>"1", "Bogdanovic"=>"2", "Jokic"=>"3");
3 krsort($kosarkasi);
4
5 foreach ($kosarkasi as $kljuc => $vrednost) {
6     echo "Kljuc = " . $kljuc . ", Vrednost = " . $vrednost;
7     echo "<br>";
8 }
9 ?>
```

Rezultat primera 35 je:

```
1 Kljuc = Teodosic, Vrednost = 1
2 Kljuc = Jokic, Vrednost = 3
3 Kljuc = Bogdanovic, Vrednost = 2
```

7 Konstante

Konstante predstavljaju jednostavne vrednosti koje se ne menjaju. Za naziv konstante važi sve što važi za naziv promenljive, s tim što se ispred naziva konstante ne stavlja znak \$. Naziv se obično piše velikim slovima. Oblast važenja konstanti je ceo program (čak i unutar funkcija). Konstante se koriste radi bolje čitljivosti i lakšeg održavanja programa.

Kreiraju se pomoću funkcije *define()*:

`define(naziv, vrednost, neosetljivost na mala i velika slova)`

Naziv i vrednost se navode kao niska. Podrazumevana vrednost trećeg argumenta funkcije je *false*, a to znači da se razlikuju mala i velika slova u nazivu konstante (kao kod promenljivih). Ako se ne želi drugačije, treći argument ne mora da se navodi.

Primer 36: Primer konstante

```
1 <?php
2 define("POZDRAV", "Zdravo, svete!");
3 echo POZDRAV;
4 ?>
```

Rezultat primera 36 je:

```
1 Zdravo, svete!
```

8 Operatori

Operatori su oznake operacija koje se izvode (sabiranje, deljenje, upoređivanje i slično), a operandi su vrednosti sa kojima se radi (konstante, promenljive, pozivi funkcija).

Operatori se dele u više grupa:

- aritmetički operatori,
- operatori dodele,
- operatori poređenja,
- operatori uvećanja i umanjenja,
- logički operatori,
- operatori nad niskama,
- operatori nad nizovima,
- bitovski operatori,
- operator izbora.

Više informacija o operatorima se može pročitati u literaturi [3].

8.1 Aritmetički operatori

Ovi operatori deluju na numeričke vrednosti i koriste se za izvođenje aritmetičkih operacija. Mogu se upotrebiti na svim numeričkim vrednostima (celi brojevi, brojevi u pokretnom zarezu i slično). Rezultat svake operacije se može sačuvati u posebnoj promenljivoj. Na primer, $\$rez = \$a + \$b$.

Operacije sabiranja, oduzimanja, množenja, deljenja i stepenovanja su iste kao i istoimene operacije u matematici. Modulo vraća ostatak pri celobrojnom deljenju. Na primer, pri celobrojnom deljenju broja 11 sa 3 dobije se 1.

U tabeli 1 su prikazani aritmetički operatori i njihovi nazivi.

Tabela 1: Aritmetički operatori

Operator	Naziv
+	Sabiranje
-	Oduzimanje
*	Množenje
/	Deljenje
%	Modulo
**	Stepenovanje

8.2 Operatori dodele

Operatori dodele deluju na numeričke vrednosti. Pomoću ovih operatora se promenljivoj sa leve strane dodeljuje vrednost koja je sa desne strane operatora. Osnovni operator dodele je `=`. Veliki broj operatora se može kombinovati sa osnovnim. Na ovaj način nastaju kompleksni operatori dodele koji skraćuju i pojednostavljaju pisanje programa.

U sledećoj tabeli su predstavljeni operatori dodele i njihovo značenje.

Tabela 2: Operatori dodele

Operator	Značenje
<code>=</code>	<code>\$x=\$y</code>
<code>+=</code>	<code>\$x=\$x+\$y</code>
<code>-=</code>	<code>\$x=\$x-\$y</code>
<code>*=</code>	<code>\$x=\$x*\$y</code>
<code>/=</code>	<code>\$x=\$x/\$y</code>
<code>%=</code>	<code>\$x=\$x%\$y</code>

8.3 Operatori poređenja

Operatori poređenja ili relacioni operatori se koriste za poređenje dve vrednosti.

Moguće je porediti i tekstualne podatke. U tom slučaju se poštuje kodni redosled. Dakle, $A < B$ i $B < a$.

Jedan od načina da se dobije logički izraz je da se uporede dve vrednosti pomoću operatora poređenja.

U tabeli 3 su prikazani operatori poređenja i rezultat koji vraćaju.

Tabela 3: Operatori poređenja

Operator	Rezultat
==	<i>True</i> ako su vrednosti koje se porede jednake
===	<i>True</i> ako su vrednosti koje se porede jednake i istog tipa
!=	<i>True</i> ako su vrednosti koje se porede različite
<>	<i>True</i> ako su vrednosti koje se porede različite
!=	<i>True</i> ako su vrednosti koje se porede različite ili različitog tipa
>	<i>True</i> ako je levi operand veći od desnog
<	<i>True</i> ako je levi operand manji od desnog
>=	<i>True</i> ako je levi operand veći ili jednak od desnog
<=	<i>True</i> ako je levi operand manji ili jednak od desnog

8.4 Operatori uvećanja i umanjenja

Operatori uvećanja (inkrementacije) i umanjenja (dekrementacije) su unarni operatori. Uvećavaju (umanjuju) vrednost operanda za jedan.

Kod prefiksnog uvećanja (umanjenja) se vrednost promenljive prvo uveća (umanji), a onda takva vrednost učestvuje u izrazu. Kod postfiksno uvećanja (umanjenja) se u izrazu koristi originalna vrednost promenljive, a nakon izračunavanja izraza se vrednost promenljive menja.

U sledećoj tabeli su prikazani operatori uvećanja i umanjenja i njihovi nazivi.

Tabela 4: Operatori uvećanja i umanjenja

Operator	Naziv
++\$x	Prefiksno uvećanje \$x
--\$x	Prefiksno umanjenje \$x
\$x++	Postfiksno uvećanje \$x
\$x--	Postfiksno umanjenje \$x

U primeru 37 je prikazana upotreba i način delovanja operatora uvećanja i umanjenja.

Primer 37: Primer upotrebe operatora uvećanja i umanjenja

```
1 <?php
2     $a=3;
3     $b=11;
4
5     echo $a++;
6     echo "<br>";
7     echo --$b;
8 ?>
```

Prvo se ispiše vrednost promenljive **\$a**, a zatim se njena vrednost uveća, dok se vrednost promenljive **\$b** prvo umanjuje, a zatim ispiše. Rezultat primera 37 je:

```
1 3
2 10
```

8.5 Logički operatori

Ono što su u govornom jeziku i, ili i ne, to su u programskom jeziku PHP logički operatori. Najčešće se koriste kada se ispituje više uslova odjednom, to jest kada se formiraju kompleksni uslovi.

U tabeli 5 su prikazani logički operatori i rezultat koji vraćaju.

Tabela 5: Logički operatori

Operator	Rezultat
!	Vraća <i>true</i> ako je operand netačan (negacija)
&&	Vraća <i>true</i> ako su oba operanda tačna (konjunkcija)
	Vraća <i>true</i> ako je barem jedan operand tačan (disjunkcija)
and	Vraća <i>true</i> ako su oba operanda tačna (konjunkcija), ima slab prioritet
or	Vraća <i>true</i> ako je barem jedan operand tačan (disjunkcija), ima slab prioritet
xor	Vraća <i>true</i> ako je tačan samo jedan operand (ekskluzivna disjunkcija), ima slab prioritet

Operatori "AND" i "OR" se ponašaju isto kao i operatori "&&" i "||", s tim što imaju mnogo niži prioritet.

8.6 Operatori nad niskama

Postoje dva operatora koja deluju samo na niske.

Operator konkatencije povezuje dve niske u jednu. Operator `.` je kompleksni operator dodele koji, takođe, spaja dve niske.

U tabeli 6 su prikazani operatori nad niskama i njihovo značenje.

Tabela 6: Operatori nad niskama

Operator	Značenje
<code>.</code>	Spaja dve niske (konkatencija)
<code>.=</code>	Nadovezuje desni operand na levi (<code>\$x.= \$y</code> ili <code>\$x=\$x.\$y</code>)

Primer 38: Primer upotrebe operatora nad niskama

```
1 <?php
2     $a="Zdravo, ";
3     $b="svete!";
4
5     echo $a.$b;
6     echo "<br>";
7     echo $a.= $b;
8 ?>
```

Rezultat primera 38 je:

```
1     Zdravo, svete!
2     Zdravo, svete!
```

8.7 Operatori nad nizovima

Pored sortiranja, moguće je izvršiti i druge operacije nad nizovima.

Kao i niske, i nizovi se mogu spojiti. To se ostvaruje pomoću unije nizova. Osim toga, nizovi se mogu porediti na različite načine.

U sledećoj tabeli su prikazani operatori nad nizovima i njihovo značenje.

Tabela 7: Operatori nad nizovima

Operator	Značenje
+	Unija dva niza
==	Vraća <i>true</i> ako nizovi imaju iste ključ/vrednost parove
===	Vraća <i>true</i> ako nizovi imaju iste ključ/vrednost parove na istim pozicija koji su istog tipa
!=	Vraća <i>true</i> ako su nizovi različiti
<>	Vraća <i>true</i> ako su nizovi različiti
!==	Vraća <i>true</i> ako nizovi nisu identični

8.8 Bitovski operatori

Bitovski operatori deluju na bitove celobrojnih operanada. Rezultat je numeričkog tipa, s tim što operatori tretiraju bitove kao logičke vrednosti. (1 kao *true*, a 0 kao *false*).

U tabeli 8 su prikazani bitovski operatori i njihovi nazivi.

Tabela 8: Bitovski operatori

Operator	Naziv
~	Negacija
&	Konjunkcija
	Disjunkcija
<<	Šiftovanje ulevo
>>	Šiftovanje udesno

8.9 Operator izbora

Ovo je jedini ternarni operator (povezuje tri operanda). Koristi se u sledećem obliku:

```
uslov ? izraz_tacno : izraz_netacno
```

Prvo se ispituje *uslov*. Ako je on zadovoljen, vrednost celog izraza je vrednost od *izraz_tacno*, a u suprotnom je vrednost od *izraz_netacno*. Može se reći da se zadaje logički uslov i dva izraza.

9 Grananja

Pri pisanju programa, u većini slučajeva je potrebno da se on izvršava na različite načine u različitim situacijama ili se u odnosu na neki uslov određuje koji deo programa će se izvršavati. U takvim slučajevima se koriste naredbe grananja.

Naredba *if* ispituje uslov i izvršava određene naredbe ako je uslov ispunjen. Sintaksa naredbe *if* izgleda ovako:

```
if ( uslov ) {  
    naredbe  
}
```

Primer 39: Primer upotrebe naredbe *if*

```
1 <?php  
2     $a=3;  
3     $b=11;  
4  
5     if($a < $b) {  
6         echo "$a je manje od $b";  
7     }  
8 ?>
```

Rezultat primera 39 je:

```
1 3 je manje od 11
```

Ukoliko je potrebno da se izvrše neke naredbe i kada uslov nije ispunjen, koristi se *if else*. Sintaksa naredbe *if else* izgleda ovako:

```
if ( uslov ) {  
    naredbe  
} else {  
    naredbe  
}
```

Primer 40: Primer upotrebe naredbe *if else*

```
1 <?php  
2     $a=3;  
3     $b=11;  
4  
5     if($a < $b) {  
6         echo "$a je manje od $b";  
7     } else {  
8         echo "$a nije manje od $b";  
9     }  
10 ?>
```

Rezultat primera 40 je:

```
1 3 je manje od 11
```

U slučaju da postoje više od dva uslova, koristi se naredba *if elseif else*. Sintaksa naredbe *if elseif else* izgleda ovako:

```
if ( uslov ) {  
    naredbe  
} elseif (uslov) {  
    naredbe  
} else {  
    naredbe  
}
```

Primer 41: Primer upotrebe naredbe *if elseif else*

```
1 <?php  
2     $a=3;  
3     $b=11;  
4  
5     if($a < $b) {  
6         echo "$a je manje od $b";  
7     } elseif ($a > $b) {  
8         echo "$a je vee od $b";  
9     } else {  
10        echo "$a je jednako $b";  
11    }  
12 ?>
```

Rezultat primera 41 je:

```
1 3 je manje od 11
```

Grananje *switch* je slično *if else* grananju. Kao ulazni parametar ima promenljivu ili izraz. Nakon ključne reči *case* pišu se moguće vrednosti promenljive ili izraza, (:) i naredbe koje će se izvršavati ako promenljiva ili izraz imaju tu vrednost. Sintaksa naredbe *switch* izgleda ovako:

```
switch ( izraz ) {  
    case vrednost1:  
        naredbe koje se izvorsavaju ako je izraz = vrednost1  
        break; //ili continue;  
    case vrednost2:  
        naredbe koje se izvorsavaju ako je izraz = vrednost2  
        break;  
    ...  
    default:  
        naredbe koje se izvorsavaju ako je izraz razlicit od svih prethodnih vrednosti  
}
```

Da se ne bi nastavilo sa izvršavanjem sledećeg slučaja, koristi se *break* ili *continue*. Pomoću *default* može se navesti deo programa (podrazumevani slučaj) koji će se izvršavati ako se nijedan drugi slučaj ne izvrši.

Primer 42: Primer upotrebe naredbe *switch*

```
1 <?php
2 $boja = "plava";
3
4 switch ($boja) {
5     case "crvena":
6         echo "Vasa omiljena boja je crvena.";
7         break;
8     case "plava":
9         echo "Vasa omiljena boja je plava.";
10        break;
11    case "zelena":
12        echo "Vasa omiljena boja je zelena.";
13        break;
14    default:
15        echo "Vasa omiljena boja nije crvena, a nije ni plava ni zelena.";
16    }
17 ?>
```

Rezultat primera 42 je:

```
1 Vasa omiljena boja je plava.
```

Napomena 4 *Naredba break prekida izvršavanje petlje ili grananja. Naredba continue preskače preostale naredbe i prelazi na sledeću iteraciju.*

Napomena 5 *Ponavljanje bloka naredbi se naziva iteracija.*

10 Petlje

Petlje izvršavaju zadate naredbe određeni broj puta. Programski jezik PHP podržava četiri petlje:

- while,
- do while,
- for,
- foreach.

Više o petljama se može pročitati u literaturi [1].

10.1 While petlja

While petlja je najjednostavnija. Izvršava naredbe sve dok je zadati uslov ispunjen (*true*). Može se desiti da se naredbe ne izvrše uopšte, a može se desiti da je uslov uvek ispunjen i da se naredbe izvršavaju beskonačan broj puta (beskonačna petlja). Sintaksa *while* petlje je sledeća:

```
while ( uslov ) {
    naredbe
}
```

Neka je potrebno ispisati cifre broja 1234. To se može uraditi pomoću *while* petlje.

Primer 43: Primer upotrebe *while* petlje

```
1 <?php
2 $broj = 1234;
3
4 while ($broj > 1) {
5     $cifra = $broj % 10;
6     echo $cifra;
7     echo "<br>";
8     $broj /= 10;
9 }
10 ?>
```

Rezultat primera 43 je:

```
1 4
2 3
3 2
4 1
```

Sve dok je broj veći od jedan, poslednja cifra se dobija tako što se uzme ostatak pri deljenju broja sa 10 (`$cifra = $broj % 10`). Cifra se ispisuje sa naredbom *echo*, a broj se podeli sa 10 (`$broj /= 10`).

10.2 *Do while* petlja

Ova petlje će uvek jednom izvršiti blok naredbi. Nakon toga će ispitati uslov i opet izvršiti naredbe ako je uslov ispunjen ili izaći iz petlje ako uslov nije ispunjen. Sintaksa *do while* petlje je:

```
do {
    naredbe
} while (uslov);
```

Treba ispisati brojeve od 1 do 5. To je lako uraditi pomoću ove petlje kao što je prikazano u primeru 44. Prvo će se ispisati broj 1, a zatim uvećati (`$broj++`). Sada se ispituje da li je novi (uvećani) broj manji ili jednak od 5 i ponavljaju se naredbe ako jeste. Sve ovo se ponavlja dok je uslov ispunjen.

Primer 44: Primer upotrebe *do while* petlje

```
1 <?php
2 $broj = 1;
3
4 do {
5     echo $broj;
6     echo "<br>";
7     $broj++;
8 } while ($broj <= 5);
9 ?>
```

Rezultat primera 44 je:

```
1 1
2 2
3 3
4 4
5 5
```

10.3 *For* petlja

For petlja ponavlja određene naredbe više puta (obično se vrednosti promenljivih menjaju u svakoj iteraciji) i koristi se kada je unapred poznato koliko puta će se izvršiti zadate naredbe. Unutar *for* naredbe zadaju se tri izraza.

- Prvi izraz se izvršava samo jednom, pre početka petlje i služi za inicijalizaciju brojača,
- Drugi izraz se izvršava pre svake iteracije. Proverava se brojač i ako je vrednost izraza *true*, petlja se nastavlja, a u suprotnom se prekida,
- Treći izraz se izvršava posle svake iteracije i menja se brojač.

Sintaksa *for* petlje izgleda ovako:

```
for (pocetni_izraz; uslov; izraz_posle_iteracije) {
    naredbe
}
```

Brojevi od 1 do 5 se mogu ispisati i pomoću *for* petlje.

Primer 45: Primer upotrebe *for* petlje

```
1 <?php
2 for ($i = 1; $i <= 5; $i++) {
3     echo $i;
4     echo "<br>";
5 }
6 ?>
```

Rezultat primera 45 je:

```
1 1
2 2
3 3
4 4
5 5
```

Pomoću *for* petlje je moguće proći kroz numerisani niz i ispisati njegove elemente kao što je prikazano u primeru 46.

Primer 46: Primer upotrebe *for* petlje i numerisanog niza

```
1 <?php
2 $kursevi = array("HTML", "CSS", "PHP", "Laravel", "Spring");
3 $duzina_niza = count($kursevi);
4
5 for ($i = 0; $i < $duzina_niza; $i++) {
6     echo $kursevi[$i];
7     echo "<br>";
8 }
9 ?>
```

Rezultat primera 46 je:

```
1 HTML
2 CSS
3 PHP
4 Laravel
5 Spring
```

10.4 *Foreach* petlja

Ova petlja služi samo za prolazak kroz asocijativne nizove, jer imaju specifičnu ključ-vrednost strukturu. Može se koristiti u dva oblika.

U prvom obliku promenljiva dobija vrednost jednog elementa. Sintaksa je sledeća:

```
foreach ($niz as $vrednost) {
    deo programa koji radi sa elementom koji ima vrednost $vrednost
}
```

Kada se menjaju elementi niza ili se želi znati pod kojim ključem se nalazi određena vrednost, koristi se sledeći oblik:

```
foreach ($niz as $kljuc => $vrednost) {
    naredbe
}
```

Sada se lako ispisuju elementi već pomenutog niza košarkaša.

Primer 47: Primer upotrebe *foreach* petlje

```
1 <?php
2 $kosarkasi = array("Teodosic"=>"1", "Bogdanovic"=>"2", "Jokic"=>"3");
3
4 foreach ($kosarkasi as $kljuc => $vrednost) {
5     echo "Kljuc = " . $kljuc . ", Vrednost = " . $vrednost;
6     echo "<br>";
7 }
8 ?>
```

Rezultat primera 47 je:

```
1 Kljuc = Teodosic, Vrednost = 1
2 Kljuc = Bogdanovic, Vrednost = 2
3 Kljuc = Jokic, Vrednost = 3
```

11 Funkcije

Deo programa koji se češće izvodi može se izdvojiti u posebnu celinu koja se poziva iz glavnog programa. PHP ima veliki broj ugrađenih funkcija, ali se mogu definisati i nove funkcije. Veća je potreba za kreiranjem funkcija što je program složeniji.

Funkcija je blok naredbi koji se može pozivati više puta u programu. Odlikuje je naziv koji treba da oslikava ono što funkcija radi. Može da ima parametre i lokalne promenljive. Funkciju je moguće pozivati iz glavnog programa, druge funkcije, pa čak može i samu sebe da poziva (rekurzija). Može da vrati vrednost, ali i ne mora.

11.1 Korisnički definisane funkcije

Funkcija se deklarise ključnom rečju *function* nakon koje se navodi naziv funkcije i lista formalnih argumenata u zagradi. Telo funkcije (njene naredbe) se navodi unutar vitičastih zagrada. Sintaksa funkcije izgleda ovako:

```
function naziv (lista argumenata) {
    naredbe
}
```

Funkcija se poziva po imenu, a u zagradi se navode vrednosti argemanata (stvarni argumenti).

Primer 48: Primer funkcije

```
1 <?php
2 function pozdrav() {
3     echo "Zdravo, svete!";
4 }
5
6 echo pozdrav(); //poziv funkcije
7 ?>
```

Rezultat primera 48 je:

```
1 Zdravo, svete!
```

11.2 Argumenti funkcije i povratne vrednosti

Preko argumenata se prenose informacije funkciji. U deklaraciji funkcije se navode formalni argumenti. Oni predstavljaju listu promenljivih koje primaju vrednosti stvarnih argumenata koji se navode pri pozivu funkcije. Formalni argumenti se posmatraju kao lokalne promenljive u koje se kopiraju vrednosti stvarnih argumenata. Dakle, iako se menjaju argumenti unutar funkcije, originalne vrednosti ostaju nepromenjene. U ovom slučaju se kaže da se argumenti prenose po vrednosti.

Ako je potrebno menjati vrednosti koje se prosleđuju funkciji, treba preneti argumente po referenci. To znači da su formalni argumenti pokazivači na stvarne argumente čije se vrednosti mogu lako menjati. Potrebno je staviti znak `&` ispred argumenata koji se prenose po referenci. U tom slučaju stvarni argumenti treba da budu promenljive.

Kada je potrebno da funkcija vrati rezultat, koristi se naredba *return*. Nakon nje se navodi vrednost koju funkcija vraća. Ova naredba označava i prekid rada funkcije.

Primer 49: Primer funkcije koja menja vrednost promenljive

```
1 <?php
2     function povecaj(&$broj, $vrednost) {
3         $broj+=$vrednost;
4     }
5
6     $br = 3;
7     povecaj($br, 8);
8     echo $br;
9 ?>
```

Rezultat primera 49 je 11.

Napomena 6 *Da bi se pristupilo globalnoj promenljivoj iz funkcije, potrebno je staviti reč `global` ispred naziva promenljive. Funkcije treba da budu fleksibilnije i nezavisnije od glavnog programa i treba izbegavati upotrebu globalnih promenljivih.*

12 Formulari

Na veb stranicama se često sreću razni formulari koje korisnik može da popuni (registracija, kupovina i slično), a služe za prikupljanje podataka. Liče na formulare na pipiru, s tim što mogu da sadrže i interaktivne elemente kao što je padajući meni.

Formular počinje etiketom *form*. Osnovni atributi formulara su *method* i *action*. Atribut *method* određuje na koji način će se podaci proslediti (*GET* ili *POST*), a *action* putanju ka stranici ili programu koji će obrađivati prosleđene podatke. Podrazumevana vrednost atributa *method* je *GET*.

I *GET* i *POST* kreiraju ključ-vrednost niz. Ključevi su nazivi prosleđenih podataka u formularu, a vrednosti su podaci koje korisnik unosi.

Primer 50: Primer formulara sa atributima

```
1 <form method="POST" action="obrada.php">
2 </form>
```

12.1 Metod GET

Podaci prosleđeni *GET* metodom su vidljivi svima, jer se prosleđuju kao deo URL putanje u obliku naziv=vrednost. Moguće je proslediti najviše 2000 karaktera. Podacima prosleđenim na ovaj način na PHP stranu se pristupa iz PHP koda pomoću niza `$_GET[]` koji je indeksiran po nazivima prosleđenih podataka.

Napomena 7 Metodom *GET* ne treba prosleđivati lozinke i druge osetljive informacije.

12.2 Metod POST

Podaci prosleđeni *POST* metodom nisu vidljivi svima i nema ograničenja na količinu podataka koja se šalje. Pristupa im se pomoću niza `$_POST[]`.

Napomena 8 Na istoj *HTML* stranici je moguće imati više formulara koji se u ovom slučaju mogu razlikovati na više načina. Jedan od njih je da svaki formular ima različitu vrednost *action* atributa. U suprotnom, drugi elementi (dugme koje šalje podatke, etiketa *form* i slično) treba da imaju različite vrednosti *name* ili *id* atributa.

12.3 Polja za unos podataka

Polja za unos podataka predstavljena su elementom sa etiketom *input*. Neki atributi ovog elementa su dati u sledećoj tabeli.

Tabela 9: Polja za unos podataka

Atribut	Značenje
type	Tip unosa koji se očekuje
value	Vrednost input polja
placeholder	Dodatno objašnjenje u obliku teksta koji se prikazuje dok se ne unese vrednost polja
required	Polje je neophodno popuniti, inače se podaci neće proslediti
disabled	Onesposobljavanje unosa

Elementom sa etiketom *label* se označava naziv polja za unos. Vrednost njegovog atributa *for* je naziv identifikatora *input* elementa na koji se labela odnosi.

Grupa polja se obuhvata u jednu celinu elementom sa etiketom *fieldset* unutar kog se elementom sa etiketom *legend* zadaje naziv grupe elemenata.

Kada se unosi tekst, vrednost atributa *type* elementa *input* je *text*. Minimalni i maksimalni broj karaktera se zadaje atributima *minlength* i *maxlength*, a vizuelna širina polja sa *size*.

Primer 51: Primer polja za unos teksta

```
1 Ime: <input type="text" name="ime" maxlength="50" size="30">
```

Za unos višelinijskog teksta se koristi element sa etiketom *textarea*. Visina i širina polja se zadaju pomoću atributa *rows* i *cols*, način prelamanja teksta na kraju linija pomoću *wrap* (moguće vrednosti su *hard* i *soft*), a maksimalni broj karaktera pomoću *maxlength*.

Primer 52: Primer polja za unos višelinijskog teksta

```
1 <textarea name="adresa" cols="50" rows="6"> <br>  
2 </textarea>
```

Kod polja za unos lozinke, vrednost atributa *type* je *password*. Može da ima attribute *maxlength* i *size*.

Primer 53: Primer polja za unos lozinke

```
1 Lozinka: <input type="password" name="lozinka" maxlength="50" size="30">
```

Za unos numeričkih vrednosti koristi se tip *number*. Atributima *min* i *max* se zadaju minimalna i maksimalna dozvoljena vrednost, a atributom *step* korak kojim su dozvoljene vrednosti rasporedene.

Primer 54: Primer polja za unos numeričkih vrednosti

```
1 <input type="number" name="broj" min="5" max="100" step="5">
```

E-mail adresa se unosi u polje koje ima tip *email*. Izgleda kao polje za unos teksta, ali ima proveru ispravnosti adrese u sintaksnom smislu. Ako adresa nije u odgovarajućem obliku, podaci iz formulara neće biti prosleđeni na obradu.

Primer 55: Primer polja za unos email adrese

```
1 <input type="email" name="emailAdresa" autocomplete>
```

Tip *checkbox* označava polja koja mogu biti uključena ili isključena (čekirana). Vrednost atributa *value* određuje vrednost koja će biti prosleđena ako je polje uključeno. Polje će biti podrazumevano uključeno kada se navede atribut *checked*.

Primer 56: Primer *checkbox* polja

```
1 <input type="checkbox" name="zapamti" value="da" checked>
```

Dugme za odabir ima tip *radio*. Da bi se odabrala samo jedna od ponuđenih vrednosti, svi radio elementi treba da imaju istu vrednost atributa *name*.

Primer 57: Primer polja tipa *radio*

```
1 <input type="radio" name="boja" value="crvena" checked /> <br>  
2 <input type="radio" name="boja" value="plava" />
```

Dugme će biti podrazumevano odabrano kada se navede atribut *checked*. Svako dugme treba da ima svoju vrednost atributa *value* da bi skripta koja prima podatke znala koje dugme je pritisnuto.

Odabir jednog od ponuđenih elemenata se može ostvariti i pomoću elementa sa etiketom *select* (padajući meni). Unutar *select* elementa se navode ponuđene vrednosti kao *option* elementi. Vrednost se zadaje atributom *value*, a između *option* etikete se navodi tekst koji će biti prikazan. Elementom *optgroup* se *option* elementi grupišu u grupe. Naziv grupe se zadaje elementom *label*.

Primer 58: Primer *select* i *optgroup* elementa

```
1 <select name="gradovi">  
2   <optgroup label="severniGradovi">  
3     <option value="SU">Subotica</option>  
4     <option value="NS">Novi Sad</option>  
5   </optgroup>  
6  
7   <optgroup label="juzniGradovi">  
8     <option value="NI">Ni</option>  
9     <option value="LE">Leskovac</option>  
10  </optgroup>  
11 </select>
```

Za slanje vrednosti formulara na dalju obradu koristi se tip *submit*. Natpis unutar elementa se zadaje atributom *value*.

Primer 59: Primer *submit* elementa

```
1 <input type="submit" value="Posalji" />
```

Vrednosti se mogu proslediti i klikom na dugme *enter*.

Dugme tipa *reset* briše sve unete vrednosti i vraća formu na podrazumevan oblik. Natpis unutar elementa se zadaje atributom *value*.

Primer 60: Primer *reset* elementa

```
1 <input type="reset" value="Ponisti" />
```

Dugme čiji je tip *button* nema predodređenu funkciju. Njegovo ponašanje treba posebno isprogramirati (povratak na prethodnu stranicu i slično). Postoje i drugi tipovi input polja kao što su *file*, *image*, *date*, *time* i drugi.

13 Validacija podataka

PHP skripta koja obrađuje podatke (skripta iz *action* atributa) treba prvo da dohvati podatke koje je korisnik uneo u formular. U zavisnosti od metoda koji je korišćen, podaci

će biti dostupni u `$_GET` ili `$_POST` polju. Članu polja pristupa se preko vrednosti atributa *name* elementa obrasca.

Neka obrazac izgleda ovako:

```
1 <form method="POST" action="obrada.php" >
2 <input type="text" name="ime" />
3 ...
4 </form>
```

Vrednosti upisanoj u polje za unos teksta će se pristupiti (u obrada.php) na sledeći način:

```
1 <?php
2 $ime = $_POST['ime'];
3 ?>
```

Slično važi za **GET** metod.

U primeru 61 je prikazana forma u koju se unose ime i prezime, a zatim se podaci šalju klikom na dugme *Posalji*.

Primer 61: Primer forme

```
1 <form method="POST" action="obradi.php" >
2 Ime: <input type="text" name="ime" size="20" > <br> <br>
3 Prezime: <input type="text" name="prezime" size="20" > <br> <br>
4 <input type="submit" value="Posalji" name="posalji" >
5 </form>
```

Unete podatke obrađuje skripta obradi.php prikazana u sledećem primeru.

Primer 62: obradi.php

```
1 <?php
2 $ime = $_POST['ime'];
3 $prezime = $_POST['prezime'];
4
5 echo "Vase ime je $ime, a prezime $prezime.";
6 ?>
```

Metod kojim su podaci poslani se proverava pomoću `$_SERVER["REQUEST_METHOD"]`.

Primer 63: Proveravanje metoda kojim su podaci poslani

```
1 <?php
2
3 if ($_SERVER["REQUEST_METHOD"] == "POST") {
4     ...
5 }
6 ?>
```

Pomoću funkcija *empty()* i *isset()* se proverava da li je neko polje popunjeno. Ako nije popunjeno, moguće je ispisati odgovarajuću poruku.

U primeru 64 je prikazana forma u koju se unose ime i prezime, a zatim se podaci šalju klikom na dugme *Posalji*. Treba proveriti da li je u formu uneto ime i ispisati odgovarajuću poruku ako nije.

Primer 64: Primer forme

```
1 <form method="POST" action="obradi.php">
2 Ime: <input type="text" name="ime" size="20"> <br> <br>
3 <span class="error">* <?php echo $imeGreska;?> </span>
4 <br><br>
5 Prezime: <input type="text" name="prezime" size="20"> <br> <br>
6 <input type="submit" value="Posalji" name="posalji">
7 </form>
```

Skripta obradi.php proverava da li je uneto ime.

Primer 65: obradi.php

```
1 <?php
2 $imeGreska = "";
3 $ime = "";
4
5 if(empty($_POST['ime'])) { //ili if(!isset($_POST['ime']))
6     $imeGreska = "Potrebno je uneti ime.";
7 } else {
8     $ime = $_POST['ime'];
9 }
10 ?>
```

Sledeći primer pokazuje kako se naručuju kolači. Kôd forme je sačuvan u naruci.html.

Primer 66: naruci.html

```
1 <html>
2 <body>
3 <b>Carstvo kolaca</b> <br> <br>
4 <form action="obrada.php" method="post">
5 Vrsta: <select name="vrsta">
6 <option>mafin</option>
7 <option>bajadera</option>
8 <option>snikers</option>
9 </select> <br> <br>
10 Kolicina: <input name="kolicina" type="text" /> <br> <br>
11 <input type="submit" value="Posalji" name="posalji">
12 </form>
13 </body>
14 </html>
```

Datoteka obrada.php, prikazana u primeru 67, obrađuje podatke iz forme.

Primer 67: obrada.php

```
1 <?php
2 $kolicina = $_POST['kolicina'];
3 $vrsta = $_POST['vrsta'];
4 echo "Narucili ste ". $kolicina . " kolaca vrste " . $vrsta . ".<br />";
5 echo "Dodjite nam opet!";
6 ?>
```

Više o formama i njihovoj validaciji se može pročitati u [4].

14 Aplikacija napravljena pomoću osnovnih elemenata programskog jezika PHP

U ovom poglavlju će biti prikazana aplikacija koja je kreirana od osnovnih elemenata programskog jezika PHP. Aplikacija je zamišljena kao kviz sa pitanjima iz opšte kulture. Predstavljena je kao sajt i sastoji se iz nekoliko povezanih stranica. Svako može da pristupi kvizu i proveri svoje znanje. Kada korisnik odgovori na pitanja, može da proveri koliko tačnih odgovora ima, kao i koji odgovori su tačni. Takođe, može ponovo da uradi kviz.

Zaglavlje naslovne stranice dato je na slici 6.



Slika 6: Zaglavlje naslovne strane

Kviz se sastoji iz deset pitanja. Za svako pitanje postoje četiri ponuđena odgovora od kojih je moguće izabrati jedno. Pitanja se nalaze unutar forme. Kôd ove stranice se nalazi u fajlu `kviz.php`.

Deo forme sa pitanjima je dat na slici 7.

1. Balet "Ohridska legenda" komponovao je:

- ☐ Kornelije Stanković
- ☐ Stevan St. Mokranjac
- ☐ Stevan Hristić
- ☐ Kosta Manojlović

2. Šta su floskule?

- ☐ Prazne reči, fraze
- ☐ Dvosmislene izjave
- ☐ Arhaizmi
- ☐ Reči negodovanja

3. Koja životinja je u dečijoj pesmici "buba lenja"?

- ☐ Cvrčak
- ☐ Lisica
- ☐ Vuk
- ☐ Slon

Slika 7: Deo forme

Forma se kreira dinamički. Pitanja se izvlače iz baze i nisu uvek ista. U nastavku je dat deo koda koji bira pitanja i pravi formu.

```
1 <?php
2     konektuj();
3     /*U bazi ima 30 pitanja, a potrebno je 10 za kviz. Bira se random broj ($br_pit) izmedju
4     1 i 21 i prvo pitanje kviza ce biti pitanje koje se u bazi nalazi pod tim rednim brojem.
5     $br_pit2 je za 10 veci od $br_pit. */
6     $br_pit = rand(1, 21);
7     $br_pit2 = $br_pit + 10;
8
9     /*Iz baze se uzimaju jedno po jedno pitanje ciji je redni broj veci ili jednak od $br_pit i
10    manji od $br_pit2. Rezultati upita se smestaju u $rezultat1 i citaju red po red.*/
11    for ($i=$br_pit; $i < $br_pit2; $i++) {
12
13        $sql1 = "SELECT * FROM 'pitanja' WHERE 'pid' = $i ";
14        $rezultat1 = mysqli_query($veza, $sql1);
15        if (mysqli_num_rows($rezultat1) > 0) {
16            while ($red1 = mysqli_fetch_assoc($rezultat1)) {
17                ?>
18
19                //Pravi se forma sa pitanjima.
20                <div class="container">
21                    <form class="forma" action="rezultat.php" method="POST">
22
23                        <br>
24                        <div class="odg">
25                            <br>
26                            /*Da bi pitanja bila numerisana od 1 do 10, potrebno je $i umanjiti za random broj
27                            $br_pit i uvecati za 1.*/
28                            <p class="odg-header"> <?php echo ($i - $br_pit + 1) .". ". $red1['pitanja']; ?> </p>
29
30                            //Ispod svakog pitanja treba ispisati odgovore koji su ponudjeni za njega.
31                            <?php
32                                $sql = "SELECT * FROM 'odgovori' WHERE 'odg_id' = $i ";
33                                $rezultat = mysqli_query($veza, $sql);
34                                if (mysqli_num_rows($rezultat) > 0) {
35
36                                    while ($red = mysqli_fetch_assoc($rezultat)) {
37                                        ?>
38
39                                    /*Ispred svakog ponudjenog odgovora treba da bude dugme za odabir. Posto za svako pitanje
40                                    treba izabrati samo jedan od cetiri ponudjena odgovora, atribut name treba da ima istu
41                                    vrednost za te ponudjene odgovore. Ta vrednosti ce biti vrednost kolone odg_id iz tabele
42                                    iz baze u kojoj se nalaze ponudjeni odgovori. Vrednost te kolone je redni broj pitanja o cijem
43                                    ➔ ponudjenom odgovoru se radi.*/
44                                    <div class="odg-block" id="radio">
45                                        <input required type="radio" oninvalid="this.setCustomValidity('Odaberite jedan od
46                                        ➔ ponudjenih odgovora!')" name="kvizproveri[<?php echo $red['odg_id']; ?>]" id="<?php echo
47                                        ➔ $i ?>" value="<?php echo $red['odgid']; ?>">
48                                        <?php echo $red['odgovori']; ?>
49                                    <br>
50                                </div>
```

Potrebno je odgovoriti na svako pitanje. Kôd koji to obezbeđuje je prikazan u sledećem redu.

```
1 oninvalid="this.setCustomValidity('Odaberite jedan od ponuenih odgovora!')"
```

Na dnu stranice se nalazi dugme Proveri. Klikom na ovo dugme se otvara nova stranica, `rezultat.php`, na kojoj piše koliko tačnih odgovara korisnik ima.

Izgled stranice `rezultat.php` je prikazan na slici 8.



Slika 8: Stranica `rezultat.php`

U nastavku je dat deo koda koji proverava koliko tačnih odgovora ima korisnik.

```

1 <?php
2 konektuj();
3 $bodovi = 0;
4 $br1 = $_POST["br_pit"]; //Redni broj (u bazi) prvog pitanja
5 $br2 = $br1 + 9; //Redni broj (u bazi) poslednjeg pitanja
6
7 $konacno = ""; //Promenljiva koja ce sadrzati poruku sa rezultatom.
8 echo $konacno;
9
10 if(isset($_POST['proveri'])) {
11     if(!empty($_POST['kvizproveri'])) {
12         $oznaceni = $_POST['kvizproveri']; //Cuva odgovore koje je korisnik odabrao.
13
14         $i = 0;
15
16         /*pid je redni broj pitanja u bazi, a odg_id je redni broj (id) ponudjenog odgovora koji je tacan
17         odgovor za dato pitanje. Sledeci upit vraca id tacnih odgovora na pitanja iz kviza.*/
18         $q1= "SELECT 'odg_id' FROM 'pitanja' WHERE 'pid'>=$br1 AND 'pid'<=$br2";
19         $rezultati = mysqli_query($veza,$q1);
20
21         //Proverava se, za svako pitanje, da li je odabrani odgovor tacan.
22         while($redovi = mysqli_fetch_array($rezultati)){
23
24             //Ako je id tacnog odgovora jednak odgovoru koji je korisnik izabrao, broj bodova se povecava.
25             if ($redovi['odg_id'] == $oznaceni[$br1+$i]) {
26                 $bodovi++;
27             }
28             $i++;
29         }
30
31         //Ispisuje se odgovarajuca poruka.
32         if($bodovi == 1)
33             $konacno = "Od 10 pitanja, odgovorili ste tano na ". $bodovi ." pitanje.";
34         else
35             $konacno = "Od 10 pitanja, odgovorili ste tano na ". $bodovi ." pitanja.";
36     }
37 }
38
39 ?>

```

Na istoj stranici se nalaze i dva dugmeta, Pokušaj ponovo i Proveri odgovore. Klikom na Pokušaj ponovo se otvara početna strana sa novim pitanjima, a klikom na Proveri odgovore se otvara stranica na kojoj su prikazana pitanja na koja je korisnik odgovorio i odgovori koje je izabrao. Tačni odgovori su zelene boje, a netačni crvene.

Primer tačnog i netačnog odgovora je prikazan na slici 9.

1. Šta su floskule?

- ☒ Prazne reči, fraze
- ☐ Dvosmislene izjave
- ☐ Arhaizmi
- ☐ Reči negodovanja

2. Koja životinja je u dečijoj pesmici "buba lenja"?

- ☐ Cvrčak
- ☒ Lisica
- ☐ Vuk
- ☐ Slon

Slika 9: Tačan i netačan odgovor

Na ovoj strani se nalazi i dugme Pokušaj ponovo. Klikom na dugme se otvara početna strana sa novim pitanjima. Kôd ove stranice se nalazi u fajlu `tacni.php`.

Dno stranice i dugme Pokušaj ponovo su prikazani na slici 10.



Copyright kviz Kvisko 2019

Slika 10: Dno stranice `tacni.php`

15 Saveti za početnike

Nakon što se savladaju osnovni pojmovi, može se napraviti aplikacija što je i prikazano u prethodnom poglavlju. Postavlja se pitanje kako pristupiti kreiranju aplikacije.

Kada se odredi šta aplikacija treba da radi, potrebno je razmisliti kako se to može realizovati i koje elemente veb aplikacija treba da ima. To se može saznati odgovaranjem na pitanja da li je potrebna baza, koliko stranica aplikacija treba da ima, da li je potrebno logovanje i slično. Prvo treba instalirati sve potrebne alate. U slučaju programskog jezika PHP, to je WampServer koji sadrži i MySQL bazu.

Važno je da aplikacija bude intuitivna i jednostavna za korišćenje. Korisnički interfejs treba da bude jednoobrazan u svim delovima aplikacije, a navigacija intuitivna i skoro neprimetna.

Za većinu aplikacija je neophodna baza u kojoj se čuvaju i (ili) iz koje se čitaju podaci. Zbog daljeg rada na aplikaciji, treba pametno napraviti bazu, to jest dobro organizovati i povezati tabele.

Ako korisnik unosi podatke, od velikog je značaja provera njihove ispravnosti. U slučaju da podaci nisu ispravni ili je došlo do problema u radu aplikacije, treba ispisati odgovorajuću poruku. Poruke su važan deo komunikacije sa korisnikom. Kada korisnik uvozi ili preuzima podatke, treba ga obavestiti kako proces napreduje.

Prvi utisak o aplikaciji pruža njen izgled. Na internetu se može naći veliki broj šablona za izgled aplikacije, ali se uvek može napraviti sopstveni.

Prilikom izrade aplikacije, lakše je kada se pravi deo po deo. Funkcionalnosti se mogu razložiti na jednostavnije celine. Može se prvo napraviti jednostavnija funkcionalnost koja se kasnije nadogradi do složene.

Problemi se uvek javljaju, ali to ne treba da obeshrabri. Na internetu se nalazi mnogo tutorijala i stranica na kojima je pisano o raznim problemima, a može se naći i literatura.

Više o pravljenju aplikacija se može pročitati u [2].

16 Napredne teme

PHP pruža mnogo mogućnosti za rad na različitim poljima. Kada se savladaju elektronske lekcije o osnovnim pojmovima programskog jezika PHP, može se preći na napredne. Neke od njih su vreme i datum, rad sa datotekama, slanje elektronske pošte, kolačići, sesije, rad sa regularnim izrazima, filteri, rukovanje greškama, izuzeci, objektno orijentisano programiranje, XML dokumenti.

Vreme i datum je jedna od lekcija koja govori o funkcijama za rad sa vremenom i datumom.

Takođe, za korisnike može biti interesantan rad sa kolačićima (eng. *cookies*) i sesijama (eng. *session*). Kolačići se upotrebljavaju za identifikaciju korisnika i pamćenje korisničkih podešavanja, a sesije kao način čuvanja tekućeg stanja aplikacije.

Programski jezik PHP sadrži funkcije za rad sa datotekama koje omogućavaju otvaranje i čitanje datoteke. Između ostalog, moguće je poslati datoteku na server, upisati podatke u nju i uključiti spoljne datoteke u kôd.

Regularni izrazi se koriste za utvrđivanje da li određeni podaci tipa string ispunjavaju date šablone.

U svakom programskom jeziku može doći do grešaka i ne može se predvideti ponašanje korisnika koje utiče na izvršavanje programa, zbog čega je bitno rukovanje greškama i izuzecima.

Podaci koje korisnik unosi nisu uvek u ispravnom obliku i mogu otežati rad stranice. Da bi se proverila ispravnost unetih podataka koriste se filteri.

Ništa manje bitan je i koncept objektno orijentisanog programiranja i rad sa objektima i klasama.

Zanimljive teme su i slanje elektronske pošte u ovom programskom jeziku i obrada XML dokumenata. XML (*eXtensible Markup Language*) je dizajniran za brzo i lako čuvanje i transport podataka.

Više o ovim temama se može pročitati u [5].

Zaključak

Elektronske lekcije o osnovnim pojmovima programskog jezika PHP bi trebalo da olakšaju njegovo savladavanje. Aplikacija opisana u radu je kreirana da bi pokazala praktičnu primenu koncepata i pojmova opisanih u elektronskim lekcijama.

U elektronskim lekcijama su obrađeni osnovni pojmovi programskog jezika PHP. Lekcije su pisane metodički sa ciljem da budu razumljive korisnicima koji ne poznaju programski jezik PHP, a žele da ga nauče. Sa savladanim osnovnim pojmovima, korisnik može da nadogradi svoje znanje, kako iz programskog jezika PHP, tako i iz drugih programskih jezika.

Svaka lekcija sadrži jedan ili više primera koji ilistruju upotrebu opisanih pojmova i koncepata. Na kraju svake lekcije sa nalazi zadatak koji služi korisnicima za proveru znanja, a moguće je i direktno pokretanje koda iz svake lekcije. Uz svaki zadatak je priložen i pomoćni kôd. Priču zaokružuje aplikacija kreirana od osnovnih pojmova programskog jezika PHP. Aplikacija je opisana u radu i predstavlja kviz iz opšte kulture.

Na internetu se može naći raznovrsna i obimna literatura o veb tehnologijama, a samim tim i o programskom jeziku PHP. Literatura je često na engleskom jeziku što usporava i otežava učenje, jer je potrebno dobro poznavanje stručnih izraza. Posebna prednost elektronskih lekcija je upravo to što su na srpskom jeziku. Sve lekcije o osnovnim pojmovima programskog jezika PHP su dostupne na platformi eŠkola veba na linku http://edusoft.matf.bg.ac.rs/eskola_veba/#/course-details/php.

Literatura

- [1] Brad Bulger, Jay Greenspan, David Wall, *MySQL/PHP Database Applications- Wiley Pub (Second Edition)*, Wiley Publishing, Inc, Indianapolis, 2004.
- [2] Miloš Dobrojević, *Kako napraviti web aplikaciju, PHP, MySQL, JavaScript*, Šprint, Beograd, 2016.
- [3] Zvaničan sajt programskog jezika PHP: <https://www.php.net/>
- [4] Sajt platforme za interaktivno učenje veb tehnologija <https://www.w3schools.com/>
<https://www.tutorialspoint.com/>
- [5] Dušan Kovačević, *Elektronske lekcije o nekim naprednim mogućnostima programskog jezika PHP*, master rad, Matematički fakultet, Beograd, 2018.