Ispitivanje uslova, iskaz IF, iskaz SWITCH, ternarni operator

Iskaz IF

- Upotreba iskaza **if** je način kontrolisanja izvršenja iskaza koji ga prate.
- Iskaz **if** procenjuje izraz koji se nalazi unutar zagrada.
- Ako je rezultat ovog izraza vrednost **true**, iskaz je izvršen.
- Ako je rezultat ovog izrara **false**, iskaz je u potpunosti izostavljen.
- Sintaksa iskaza if:

```
if(uslov){ kod koji se izvršava ako je if uslov tačan}
```

```php
<?php
$a=5;

if($a<10){
    print "broj $a je manji od broja 10";
}

?>
```

broj 5 je manji od broja 10

Upotreba odredbe else u uskazu if

Kada se koristi iskaz if, da bi se definisao alternativni oblik koda koji treba da bude izvršen ako je iskaz koji se testira procenjen na false, treba dodati odredbu else u iskaz if.

```php
<?php
$a=5;
if($a>10){
    print "broj $a je veci od broja 10";
}
//ako uslov u if-u nije tacan izvrsava se deo koda else
else{
    print "broj $a je manji od broja 10";
}
?>
```

broj 5 je manji od broja 10

- Može da se upotrebi odredba **if.....elseif.....else** za testiranje više izraza.
- Uvek se počinje iskazom **if**, zatim koliko god je potrebno iskaza **elseif** i na kraju iskaz **else**, ako ni jedan od prethodnih iskaza nije tačan:

```php
<?php
$mesec="Novembar";

if($mesec=="Decembar" or $mesec=="Januar" or $mesec=="Februar"){
    print "Sada je zima";
}
elseif($mesec=="Mart" or $mesec=="April" or $mesec=="Maj"){
    print "Sada je prolece";
}
elseif($mesec=="Jun" or $mesec=="Juli" or $mesec=="Avgust"){
    print "Sada je leto";
}
elseif($mesec=="Septembar" or $mesec=="Oktobar" or $mesec=="Novembar"){
    print "Sada je jesen";
}
else{
    print "mesec($mesec) je pogresno napisan!!!";
}
?>
```

Sada je jesen

Iskaz switch

- Iskaz **switch** je alternativni način menjanja toka, zasnovan na proceni izraza.
- Kada se upotrebi iskaz **if**, zajedno sa iskazom **elseif**, može se proceniti više izraza.
- Iskaz **switch** procenjuje samo jedan izraz u listi izraza nakon što ga selektuje na osnovu specifičnog dela podudarnog koda.

Sledi primer iskaza switch:

```php
<?php
$omiljena_boja="plava";
switch($omiljena_boja){
    case "crvena":
        print "Moja omiljena boja je $omiljena_boja";
        break;
    case "zelena":
        print "Moja omiljena boja je $omiljena_boja";
        break;
    case "plava":
        print "Moja omiljena boja je $omiljena_boja";
        break;
    default:
        print "Moja omiljena boja nije ni plava, ni crvena, ni zelena";
        break;
}
?>
```

Moja omiljena boja je plava

Iskaz switch

- Promenljivoj **$omiljena_boja** smo dodelili vrednost **plava**.
- Iskaz **switch** koristi ovu promenljivu kao svoj izraz.
- Prvi izraz **case** testira jednakost između vrednosti **crvena** i vrednosti promenljive **$omiljena_boja**.
- U prvom izrazu **case** ne postoji podudarnost između **crvena** i promenljive **$omiljena_boja**.
- Pošto ne postoji podudarnost u prvom izrazu **case**, izvršenje skripte se pomera na sledeći izraz **case**.
- Kada skripta dođe do dela koda gde se izraz **case** podudara sa promenljivom **$omiljena_boja**, izvršava se blok koda za taj izraz **case**.
- Iskaz **break** završava proces **switch**.
- Izraz **default** obezbeđuje standardnu akciju u slučaju da ni jedan od prethodnih **case** izraza nije procenjen kao **true**.

Da bi se istakla važnost iskaza **break**, pogledaće se sledeći primer:

```php
<?php
$omiljena_boja="plava";
switch($omiljena_boja){
    case "crvena":
        print "Moja omiljena boja je $omiljena_boja";
        break;
    case "zelena":
        print "Moja omiljena boja je $omiljena_boja";
        break;
    case "plava":
        print "Moja omiljena boja je $omiljena_boja";
        //namerno smo uklonili break; sto znaci da se proces switch ne zavrsava!!!
    default:
        print "Moja omiljena boja nije ni plava, ni crvena, ni zelena";
        break;
}
?>
```

Pošto smo uklonili **break** izvršava se blok koda sve do sledećeg **break**.

Moja omiljena boja je plavaMoja omiljena boja nije ni plava, ni crvena, ni zelena

Još jedan primer iskaza switch:

```php
<?php
$mesec="novembar";
switch($mesec){
    case "decembar" :
    case "januar":
    case "februar":
        print "sada je $mesec, i zima je.";
        break;
    case "mart":
    case "april":
    case "maj":
        print "sada je $mesec, i prolece je.";
        break;
    case "jun" :
    case "juli":
    case "avgust":
        print "sada je $mesec, i leto je.";
        break;
    case "septembar" :
    case "oktobar":
    case "novembar":
        print "sada je $mesec, i jesen je.";
        break;
    default:
        print "pogresto ste uneli mesec: $mesec.";
}
?>
```

sada je novembar, i jesen je.

Upotreba operatora ?:

- **Ternarni operator** ili **?:** je sličan iskazu **if**, osim što vraća vrednost izvedenu iz jednog od dva izraza koja su odvojena dvotačkom.
- Izraz koji je upotrebljen za generisanje vraćene vrednosti zavisi od rezultata izraza testa:

(izraz) ? vraća ako je izraz true : vraća ako je izraz false

```php
<?php
$ocena1=8;
($ocena1>5) ? print "Ispit je polozen sa ocenom $ocena1" : print "Pali ste ispit";
print "<BR>";
$ocena2=5;
($ocena2>5) ? print "Ispit je polozen sa ocenom $ocena2" : print "Pali ste ispit";
?>
```

Ispit je polozen sa ocenom 8
Pali ste ispit

Implementacija petlji, iskaz for, while, do while

Implementacija petlji
iskaz **while**

- Skript može da odlučuje koliko puta će biti izvršen blok koda.
- Iskazi petlje su specifično dizajnirani da omoguće da se izvrše zadaci koji se ponavljaju, zato što nastavljaju petlju dok specifičan uslov nije ispunjen ili dok eksplicitno se ne izabere izlazak iz petlje.
- Iskaz **while** je sličan po strukturi **if**, ali ima mogućnost ponavljanja.

while(uslov){ uradi nesto}

```php
<?php
$a=0;
while($a<3){//petlja ce se izvrsavati sve dok je promenljiva $a<3
    print "sada je a=$a";
    print "<BR>";
    $a++; //povecava vrednost a za 1 posle ove polse 6. linije koda
}
?>
```

sada je a=0
sada je a=1
sada je a=2

- Iskaz **do while** izgleda kao iskaz **while** okrenut naopačke.
- Glavna razlika između ova dva iskaza proističe iz činjenice da se blok koda izvršava pre, a ne posle testa istinitosti.

do {kod koji se izvrsava}

while(uslov)

- Ovaj tip iskaza je koristan kada hoćemo da blok koda bude izvršen **bar jednom**, čak i ako je iskaz **while** ocenjen kao **false**.

```php
<?php
$a=0;
do{
    print "sada je a=$a";
    $a++;
}while($a>10); //posto je $a=0 vidimo da je uslov false
?>
```

sada je a=0

Vidi se da je blok koda izvršen jednom i ako je uslov **false**.

1. Ispisati brojeve od 2 do 10 koristeći **while** petlju.

2. Ispisati stepene broja 5 do 5. stepena (5, 5*5, 5*5*5,.....) koristeći **do while** petlju.

1.

```php
<?php
$a=1;
$b=2;
while($a<6){
    print "$b";
    print "<BR>";
    $a++;
    $b+=2;
}
?>
```

može i ovako

```php
<?php
$a=2;
while($a<11){
    print "$a";
    print "<BR>";
    $a+=2;
}
?>
```

2
4
6
8
10

2.

```php
<?php
$b=1;
$a=5;
do{
    print "5 na $b. stepen = $a";
    print "<BR>";
    $a=$a*5;
    $b++;
}while($b<6);
?>
```

5 na 1. stepen = 5
5 na 2. stepen = 25
5 na 3. stepen = 125
5 na 4. stepen = 625
5 na 5. stepen = 3125

Implementacija petlji
iskaz **for**

- Sve što se izvršava pomoću iskaza **for** može da se izvrši i pomoću iskaza **while**. Međutim, iskaz **for** je često pogodniji metod postizanja istog efekta.
- Koristeći iskaz **for**, može da se izvrši serija događaja, u jednoj liniji koda.
- To će omogućiti kompaktniji kod i umanjuje opasnost da se zaboravi povećavanje promenljive brojač čime se kreira **beskonačna petlja**.

- Sledi primer **beskonačne petlje** kako bi razumeli šta znači beskonačna petlja.
- Beskonačne petlje su, kao što i sam naziv govori, petlje koje se pokreću bez granica.
- Ako se petlja izvršava beskonačno i skript će se pokretati beskonačno.

```php
<?php
$a=0;
while($a<5){
    print "Vrednost promenljie a je = $a";
    print "<BR>";
    //$a++ namerno izostavljamo, sto ce dovesti do kreiranja beskonacne petlje
}
?>
```

Vrednost promenljie a je = 0
Vrednost promenljie a je = 0
Vrednost promenljie a je = 0
Vrednost promenljie a je = 0
Vrednost promenljie a je = 0
Vrednost promenljie a je = 0

Sintaksa iskaza:

for (dodeljivanje vrednosti brojaču; uslov koji se odnosi na brojač; promena vrednosti brojača)

Primer kako bi bolje razumeli for petlju:

```php
<?php
//Ispistati brojeve od 1 do 5 koristeci for petlju

for($a=1;$a<=5;$a++){ //$a je brojac; uslov do kog borjac ide; povecanje brojaca
    print $a;
    print "<BR>";
}
?>
```

1
2
3
4
5

Primer. Koristeći petlju **for** ispisati proizvod broja 2 sa brojevima od 1 do 20:

```php
<?php
for($a=1;$a<=20;$a++){
    $b=2;
    print "$a * $b = ".($a*$b);
    print "<BR>";
}
?>
```

1 * 2 = 2
2 * 2 = 4
3 * 2 = 6
4 * 2 = 8
5 * 2 = 10
6 * 2 = 12
7 * 2 = 14
8 * 2 = 16
9 * 2 = 18
10 * 2 = 20
11 * 2 = 22
12 * 2 = 24
13 * 2 = 26
14 * 2 = 28
15 * 2 = 30
16 * 2 = 32
17 * 2 = 34
18 * 2 = 36
19 * 2 = 38
20 * 2 = 40

Primer. Koristeći petlju **for** ispisati sve brojeve koji su deljivi sa 5 od 1 do 15, sa porukom “broj je deljiv sa 5 i on je: ”. Takođe, ispisati brojeve koji nisu deljivi sa 5 sa porukom “broj nije deljiv sa 5 i on je: ”. Ispis ide od broja 1 do 15.

```php
<?php
for($a=1;$a<=15;$a++){
    if($a%5==0){
        print "broj je deljiv sa 5 i on je: $a";
        print "<BR>";
    }
    else{
        print "broj nije deljiv sa 5 i on je: $a";
        print "<BR>";
    }
}
?>
```

broj nije deljiv sa 5 i on je: 1
broj nije deljiv sa 5 i on je: 2
broj nije deljiv sa 5 i on je: 3
broj nije deljiv sa 5 i on je: 4
broj je deljiv sa 5 i on je: 5
broj nije deljiv sa 5 i on je: 6
broj nije deljiv sa 5 i on je: 7
broj nije deljiv sa 5 i on je: 8
broj nije deljiv sa 5 i on je: 9
broj je deljiv sa 5 i on je: 10
broj nije deljiv sa 5 i on je: 11
broj nije deljiv sa 5 i on je: 12
broj nije deljiv sa 5 i on je: 13
broj nije deljiv sa 5 i on je: 14
broj je deljiv sa 5 i on je: 15

Iskazi break, continue.
Ugnježdavanje petlji

Pokretanje petlje pomoću iskaza break

- Iskazi **while** i **for** koriste ugrađeni izraz testa pomoću kojeg može da se završi petlja.
- Međutim, iskaz **break** omogućava da se zaustavi petlja na osnovu rezultata dodatnih testova. To može da nas obezbedi od pravljenja greške.

```php
<?php
 $a=0;
for($b=0;$b<10;$b++){
    if($b==5){
        break;  //izlaz iz petlje u trenutku kada je vrednost $b=5;
    }
    else{
        print "$a";
        print "<BR>";
    }
    $a++;
}
?>
```

```
0
1
2
3
4
```

Dodat je u **for** petlji uslov **if** tako da kada **$b** dođe do vrednosti 5 sledi **break**, što znači da skript izlazi iz petlje **for**.

Sve dok **$b** nema vrednost **5** izvršava se **else**.

Izostavljanje primera pomoću iskaza countinue

- Iskaz **continue** završava izvršenje aktuelne iteracije, ali ne dovodi do završavanja petlje kao celine. Umesto toga, odmah započinje sledeća iteracija.

```php
<?php
/*
 * Ispisati kolicnik broja 10 sa brojevima
 * od -5 do 5, ali tako da izostavite deljenje sa 0
*/
$deljenik=10;
for($delilac=-5;$delilac<=5;$delilac++){
    if($delilac==0){
        continue;
    }
    else{
        print "$deljenik : $delilac = ".($deljenik/$delilac);
        print "<BR>";
    }
}
?>
```

10 : -5 = -2
10 : -4 = -2.5
10 : -3 = -3.3333333333333
10 : -2 = -5
10 : -1 = -10
10 : 1 = 10
10 : 2 = 5
10 : 3 = 3.3333333333333
10 : 4 = 2.5
10 : 5 = 2

Vidimo da je u **for** petlji preskočen korak kada delilac ima vrednost 0. Takođe vidimo da iskaz **countinue** ne izlazi iz petlje već samo preskače korak za razliku od iskaza **break**.

Ugnježdavanje petlji

- Petlje mogu da sadrže druge iskaze petlji, sve dok je logika validna i dok su petlje dobro napisane.
- Kombinacija takvih iskaza je posebno korisna kada se koriste dinamički kreirane HTML tabele.
- Sledeći programski kod upotrebiće dva iskaza for za štampanje tablice množenja u pretraživaču.

```php
<?php
print "<table style='border: 1px solid #000000'>";
for($x=1;$x<=10;$x++){
    print "<tr>";
    for($y=1;$y<=10;$y++){
        print "<td style='border: 1px solid #000000; width: 25px; text-align: center'>";
        print ($x*$y);
        print "</td>";
    }
    print "</tr>";
}
print "</table>";
```

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100

- 2. linija koda određuje debljinu okvira tabele (1px), i boju (solid #000000). Prva **for** petlja formira redove, dok druga formira polja u tabeli sa proizvodom brojeva.
- **Objašnjenje**: krećemo od 1. for petlje, $x=1, ulazimo u 2. petlju $y=1, pravi se prvo polje u prvom redu sa vrednošću $x*$y= 1*1=1, zatim smo i dalje u 2. petlji, a vrednost $y je sada 2, pravi se 2 polje u prvom redu sa vrednošću 1*2=2 i sve tako do $y=10, posle čega se vraćamo u prvu petlju gde je $x sada 2, prva petlja izvrsava prelaz u novi red, zatim ulazimo u 2. petlju i tako dalje sve dok se ne završi rad ovih ugnježdenih petlji.
- 6. linija koda **width:25px** označava širinu polja, a **text-align: center** da tekst u polju bude centriran.