Lekcja 41. (kl. II. PR)

Temat: Stałe i zmienne, czyli typy danych w języku PHP.

Cele lekcji:

– posługiwanie się różnymi typami danych w PHP,
– posługiwanie się niektórymi operatorami PHP

Uczeń:

– dobiera typy zmiennych w skrypcie PHP;
– definiuje i używa stałych w PHP;
– używa operatorów w prostych skryptach PHP.

Podręcznik str. 258 – Informatyka. 2. PR, Operon

Przebieg lekcji:

1. Zapoznanie się z celami lekcji.
2. Zmienne w PHP.
3. Definiowanie i użycie stałych.
4. Operatory.
5. Ćwiczenia praktyczne.

Film: https://www.youtube.com/watch?v=tD0Q5QwoQJI

Kurs: https://phpkurs.pl/

Zadania do wykonania – podręcznik str. 262

```
<?php
    echo "To jest test";
?>

<?php echo "To jest test" ?>
```

Komentarze:

```
<?php

   echo "To jest test komentarzy"; // Ta metoda znana jest z języków C/C++

   echo "A to drugi test"; # A ta z powłok Uniksowych

?>


<?php

   echo "Test komentarzy"; /* Tu jest początek komentarza
   tu dalej trwa
```

```
   a tu się kończy */

?>
```

Zmienne:

```
<?php

   $nazwa = 1; // Zmiennej "nazwa" przypisywana jest wartość liczbowa 1

   $druga_nazwa = "Tekst"; // Zmiennej "druga_nazwa" przypisany jest ciąg znaków "Tekst"

   $trzecia_nazwa = $nazwa; // Zmiennej "trzecia_nazwa" przypisywana
                            //jest wartość zmiennej "nazwa"

   echo "To jest $druga_nazwa"; // Powinien wyświetlić się napis "To jest Tekst"

   echo '$druga_nazwa'; // Powinien wyświetlić się napis "$druga_nazwa"

   echo $nazwa; // Powinna wyświetlić się cyfra 1

?>
```

Typy zmiennych

- liczby całkowite (integer)
- liczby rzeczywiste (double)
- ciągi (string)
- tablice (array)
- obiekty (object)

Dodatkowo PHP potrafi konwertować zmienne całkowite zapisane w różnych formatach liczbowych.

Przykład 2.5. Formaty liczbowe

```
<?php
$a = 1234; # liczba dziesiętna
$a = -123; # liczba ujemna
$a = 0123; # liczba ósemkowa (równoznaczne z dziesiętnym 83)
$a = 0x12; # liczba szesnastkowa (równoznaczne z dziesiętnym 18)
?>
```

Zmiana typu

Zazwyczaj nie jest konieczne określenie typu zmiennej – PHP sam to ustala, zależnie od kontekstu.

Przykład. Zmiana typu zmiennej

```
<?php
$blah = "0";  // $blah jest ciągiem (ASCII 48)
$blah++;      // $blah jest ciągiem "1" (ASCII 49)
$blah += 1;   // $blah jest teraz wartością całkowitą (2)
$blah = $foo + 1.3;  // $blah jest wartością rzeczywistą (1.3)
$blah = 5 + "10 Malutkich Świnek"; // $blah jest wartością całkowitą (15)
$blah = 5 + "10 Małych Świń";     // $blah jest wartością całkowitą (15)
?>
```

Rzutowanie typów

```
<?php
$liczba_calkowita = 10;

$liczba_rzeczywista = (real) $liczba_calkowita;
?>
```

Dozwolone typy rzutowań

- (int), (integer) – rzutuj do typu całkowitego
- (real), (double), (float) – rzutuj do typu rzeczywistego
- (string) – rzutuj do ciągu
- (array) – rzutuj do tablicy
- (object) – rzutuj do obiektu

Drugim sposobem, trwałym, jest użycie funkcji `settype`. Funkcja ta pobiera 2 argumenty. Pierwszym jest nazwa zmiennej do ustalenia typu, a drugim ciąg określający nowy typ zmiennej.

Dopuszczalne argumenty funkcji settype

- „integer”
- „double”
- „string”
- „array”
- „object”

Funkcja zwraca wartość „true” gdy wszystko poszło pomyślnie. W przeciwnym razie zwracana jest wartość „false”.

Przykład. Przykład użycia funkcji settype

```
<?php

   $zmienna = 10.3;

   echo "$zmienna <br>"; // Wyświetlona wartość to "10.3"

   settype($zmienna, "integer");

   echo "$zmienna <br>"; // Wyświetlona wartość to "10"

?>
```

Definiowanie stałych

```
<?php
  define("STALA", "Hello world.");
  echo STALA; // Wyświetla "Hello world."
?>
```

Operatory arytmetyczne

Operatory te każdy powinien pamiętać z podstawówki

Przykład	Nazwa	Wynik
$a + $b	Dodawanie	Suma $a i $b.
$a – $b	Odejmowanie	Różnica $a i $b.
$a * $b	Mnożenie	Iloczyn $a i $b.
$a / $b	Dzielenie	Iloraz $a i $b (bez reszty).
$a % $b	Modulo	Reszta z dzielenia $a przez $b.

Operator przypisania

Podstawowym operatorem przypisania jest symbol ‚=’. Oczywiście nie oznacza on ‚jest równe’. Wyrażenie `$b = 5` oznacza, że zmienna $b przyjmuje wartość równą 5. Zmiennej można przypisać także wartość innej zmiennej: `$b = 5; $a = $b;` – zmienna $a przyjmie wartość `5`.

Zmiennym można przypisywać nie tylko konkretne wartości, ale też wartości innych zmiennych. Wartości te można przypisywać kaskadowo, przy czym wartości przypisywane będą od prawej do lewej, np.:

```
<?php
$nazwa = $inna_nazwa = $trzecia_nazwa = 5;
?>
```

W tym wypadku wszystkim zmiennym zostanie przypisana wartość 5. Operator przypisania można łączyć z operatorami arytmetycznymi i operatorem łączenia ciągów:

Przykład	Wynik
$a += 2	Do zmiennej $a dodane zostanie 2
$a -= 2	Od zmiennej $a odjęte zostanie 2
$a *= 2	Zmienna $a zostanie pomnożona przez 2
$a /= 2	Zmienna $a dodane podzielona przez 2
$a %= 2	Zmienna $a przyjmie wartość reszty z dzielenia $a przez 2
$a .= ” dalszy ciąg”	Do zmiennej $a na końcu dodany zostanie ciąg ” dalszy ciąg”

Operatory operacji bitowych

Operatory operacji bitowych pozwalają na przestawianie pojedyńczych bitów zmiennych. Poniższa tabelka przeznaczona jest dla osób, które miały już jakąkolwiek styczność z operacjami na bitach.

Przykład	Nazwa	Wynik
$a & $b	AND	Ustawiane są bity które są ustawione w obu zmiennych.
$a | $b	OR	Ustawiane są bity, które są ustawione w jednej lub drugiej zmiennej.
$a ^ $b	XOR	Ustawiane są bity, które są ustawione w jednej lub drugiej zmiennej, ale nie w obu.
~ $a	NOT	Inwerter – ustawiane są bity które nie są ustawione w zmiennej $a i odwrotnie.
$a << $b	Przesunięcie w lewo	Przesuń bity z $a $b-razy w lewo (każdy krok oznacza pomnożenie przez 2)
$a >> $b	Przesunięcie w prawo	Przesuń bity z $a $b-razy w prawo (każdy krok oznacza podzielenie przez 2)

Operatory porównania

Operatory porównania są niezbędne do korzystania z instrukcji warunkowych (jeśli coś to zrób coś). Zwracają one wartość TRUE (prawda – 1) lub FALSE (fałsz – 0).

Przykład	Nazwa	Wynik
$a == $b	Równy	Prawda jeśli $a jest równe $b.
$a === $b	Identyczny	Prawda jeśli $a jest równe $b i są tego samego typu. (tylko PHP4)
$a != $b	Nie równe	Prawda jeśli $a nie jest równe $b.
$a !== $b	Nie identyczny	Prawda jeśli $a nie jest równe $b lub nie są tego samego typu. (tylko PHP4)
$a < $b	Mniejsze	Prawda jeśli $a jest mniejsze niż $b.
$a > $b	Większe	Prawda jeśli $a jest większe niż $b.
$a <= $b	Mniejsze lub równe	Prawda jeśli $a jest mniejsze lub równe $b.
$a >= $b	Większe lub równe	Prawda jeśli $a jest większe lub równe $b.

Operator kontroli błędów

Operator kontroli błędów (‚@') powoduje, że wyrażenie przed którym postawiono ten znak nie spowoduje wyświetlenia się jakiegokolwiek błędu lub ostrzeżenia.

Przykład 3.1. Przykład użycia operatora kontroli błędów

```
<?php
/* Jeden z najczęstszych błędów SQL (za dużo o jeden apostrof) */
$res = @mysql_query ("select nazwa, kod from 'lista") or
    die ("Zapytanie się nie powiodło: błąd to '$php_errormsg'");
?>
```

Operator wywołania

Operator ten służy do uruchamiania zewnętrznych programów lub poleceń powłoki. Wystarczy wpisać polecenie pomiędzy znaki odwróconego apostrofu (‚`') aby zostało ono wykonane.

Przykład 3.2. Przykład użycia operatora wywołania

```
<?php

  $wynik = `ls -l /home/`;

  echo $wynik;

?>
```

Po uruchomieniu tego skryptu wyświetlona zostanie zawartość katalogu /home na serwerze.

Operatory inkrementacji i dekrementacji

Operatory te występują w większości języków programowania. Służą one do zmniejszenia lub zwiększenia wartości danej zmiennej o 1. Każdy operator można stosować na 2 sposoby:
preinkrementacja/predekrementacja – najpierw wartość zmiennej zostanie zmieniona, a później zwrócona, lub postinkrementacji/postdekrementacji – najpierw zostanie zwrócona wartość zmiennej, a następnie wartość zmiennej zostanie zmieniona.

Przykład	Nazwa	Wynik
++$a	Preinkrementacja	Zwiększa $a o jeden, a następnie zwraca $a.
$a++	Postinkrementacja	Zwraca $a, a następnie zwiększa $a o jeden.
–$a	Predekrementacja	Zmniejsza $a o jeden, po czym zwraca $a.
$a–	Postdekrementacja	Zwraca $a, po czym zmniejsza $a o jeden.

Przykład. Przykład funkcjonowania inkrementacji i dekrementacji

```
<?php
echo "Postinkrementacja";
$a = 5;
echo "Powinno być 5: " . $a++ . "\n";
echo "Powinno być 6: " . $a   . "\n";

echo "Preinkrementacja";
$a = 5;
echo "Powinno być 6: " . ++$a . "\n";
echo "Powinno być 6: " . $a   . "\n";

echo "Postdekrementacja";
$a = 5;
echo "Powinno być 5: " . $a-- . "\n";
echo "Powinno być 4: " . $a   . "\n";

echo "Predekrementacja";
$a = 5;
echo "Powinno być 4: " . --$a . "\n";
echo "Powinno być 4: " . $a   . "\n";
?>
```

Operatory logiczne

Operatory logiczne służą do budowania bardziej skomplikowanych instrukcji warunkowych – do łączenia kilku warunków w jednej instrukcji.

Przykład	Nazwa	Wynik
$a && $b	AND	Prawda, jeśli $a i $b są prawdą
$a || $b	OR	Prawda, jeśli $a lub $b są prawdą
! $a	NOT	Prawda, jeśli $a nie jest prawdą

Operator ciągu

Operator ciągu (‚.’ – kropka) służy do łączenia kilku ciągów w jedną całość.

Przykład. Przykład użycia operatora ciągu

```
<?php

   $zmienna1 = "Wartość zmiennej 'zmienna2' to";

   $zmienna2 = 5;

   echo $zmienna1." ".$zmienna2; // Powinno się wyświetlić:
```

```
                                    // "Wartość zmiennej 'zmienna2' to 5"

?>
```

Jak widać na tym przykładzie, aby użyć niektórych znaków (między innymi cudzysłowów jeśli ciąg podany jest cudzysłowach, znaków dolara jeśli nie chcemy aby został potraktowany jako zmienna) trzeba zamienić go na tzw. sekwencję escape, to znaczy wstawić przed nim znak *backslash* – \.