PHP

Cz.2

Składnia

Składnia PHP jest podobna (prawie identyczna) jak składnia C++.

Są oczywiście między tymi językami różnice, ale jeśli ktoś miał do czynienia z C++ to z PHP bardzo szybko się ‘zaprzyjaźni’.

Oddzielanie instrukcji

Jedną z głównych zasad języka PHP jest umieszczenie na końcu każdej instrukcji (niekoniecznie linii) znaku średnika (‚;').

Można go pominąć tylko jeśli w danym miejscu następuje przejście do trybu HTML, a więc po danej linii następuje symbol przejścia do trybu HTML. Ale generalnie znak średnika jest wymagany.

```
<?php
    echo "To jest test";                    //wersja gdzie średnik był konieczny
?>

<?php echo "To jest test" ?>                //wersja bez średnika
```

Komentarze

Czasem zachodzi potrzeba oznaczenia czegoś w kodzie, dla kogoś innego czy nawet dla siebie samego. Wtedy można skorzystać z jednej z kilku metod oznaczania, dzięki którym parser PHP będzie wiedział, że dany tekst nie jest częścią skryptu i można go zignorować.

Komentarze przydają się także do tymczasowego „wyłączania” niektórych linii kodu.

```
<?php
    echo "To jest test komentarzy"; // Ta metoda znana jest z języków C/C++
    echo "A to drugi test"; # A ta z powłok Uniksowych
?>
```

Komentarze wieloliniowe

```
<?php
    echo "Test komentarzy"; /* Tu jest początek komentarza
        …
         a tu się kończy */
?>
```

Zmienne

Zmienna jest to identyfikator znakowy, któremu przypisano jakąś wartość.

W języku PHP zmienne oznacza się za pomocą znaku dolara (‚$') przed wspomnianym identyfikatorem. Obsługa zmiennych w PHP jest uproszczona do minimum.

W „dużych" językach programowania zmienne trzeba najpierw inicjować (przy czym z góry trzeba określić typ zmiennej), zmienne tekstowe muszą mieć z góry ustalony rozmiar itp. W PHP nie jest to konieczne. Zmienna jest inicjalizowana (to znaczy rezerwowany jest dla niej pewien obszar w pamięci) przy pierwszym jej użyciu.

Nazwy zmiennych muszą zaczynać się od litery (dużej lub małej) lub „underscore" (dolna kreska – ‚_') a dalej mogą się składać z dowolnej ilości liter, cyfr i znaków o kodzie ASCII powyżej 127. Przy nazwach zmiennych respektowana jest wielkość znaków – zmienne $Test i $test to dwie różne zmienne.

Zmienne – przykład:

```
<?php
  $nazwa = 1; // Zmiennej "nazwa" przypisywana jest wartość liczbowa 1
  $d_nazwa = "Tekst"; // Zmiennej "d_nazwa" przypisany jest ciąg znaków "Tekst"
  $t_nazwa = $nazwa; // Zmiennej "t_nazwa" przypisywana jest wartość zmiennej
                                                                    „nazwa"
  echo "To jest $d_nazwa"; // Powinien wyświetlić się napis "To jest Tekst"
  echo '$d_nazwa'; // Powinien wyświetlić się napis "$d_nazwa"
  echo $nazwa; // Powinna wyświetlić się cyfra 1
?>
```

Parametr dla polecenia echo można podawać zarówno w cudzysłowach jak i apostrofach. Jednak te parametry nie są sobie równoznaczne. W przypadku cudzysłowów zmienne zawarte między nimi są zamieniane na ich wartość, a w przypadku apostrofów zmienna pozostaje swoją nazwą.

Typy zmienych

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

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

Zmienne - 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.

```
<?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)
?>
```

Podczas przypisywania zmiennej nowej wartości, poprzednia wartość jest oczywiście zamazywana. W takim przypadku typ zmiennej ustalany jest od nowa.

Typy – Rzutowanie

Jeśli zachodzi potrzeba zmiany typu lub PHP błędnie rozpoznaje typ, to można tego dokonać za pomocą rzutowania (cast – efekt jest jednorazowy) lub za pomocą funkcji settype (efekt trwały).

Rzutowanie typów odbywa się przez podanie nowego typu w nawiasie przed zmienną lub wartością, której typ chcemy zmienić.

```
<?php
    $liczba_calkowita = 10;
    $liczba_rzeczywista = (real) $liczba_calkowita;
?>
```

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"
?>
```

Predefiniowane zmienne

W każdym skrypcie PHP dostępne jest kilka zmiennych, których wartość jest ustalana na podstawie zmiennych środowiskowych serwera WWW. Dostępne są jak zwykłe zmienne – ze znakiem dolara przed nazwą.

Zmienne ustawiane przez serwer WWW:

GATEWAY_INTERFACE

Informacja o specyfikacji CGI używanej przez serwer, np. ‚CGI/1.1’.

SERVER_NAME

Nazwa hosta serwera na którym skrypt jest uruchamiany. Jeśli skrypt pracuje na wirtualnym hoście, to zmienna przyjmie jako wartość nazwę wirtualnego hosta.

Zmienne ustawiane przez serwer WWW:

SERVER_SOFTWARE

Ciąg identyfikujący serwera podawany przy odpowiadaniu na zapytania.

SERVER_PROTOCOL

Nazwa i numer wersji protokołu za pomocą którego wysłano zapytanie o stronę, np. ‚HTTP/1.0’;

REQUEST_METHOD

Metoda zapytania użyta do uzyskania dostępu do strony, np. ‚GET’, ‚HEAD’, ‚POST’, ‚PUT’.

QUERY_STRING

Ciąg zapytania (jeśli takowy istnieje) za pomocą którego połączono się ze stroną.

Zmienne ustawiane przez serwer WWW:

DOCUMENT_ROOT

Katalog główny drzewa dokumentów spod którego skrypt jest wykonywany – jest to ustawienie z pliku konfiguracyjnego serwera.

HTTP_ACCEPT

Nagłówek z aktualnego zapytania, jeśli taki istnieje.

HTTP_ACCEPT_CHARSET

Zawartość nagłówka „Accept-Charset” z aktualnego zapytania, jeśli taki istnieje, np. ‚iso-8859-1,*,utf-8’.

HTTP_ENCODING

Zawartość nagłówka „Accept-Encoding” z aktualnego zapytania, jeśli taki istnieje, np. ‚gzip’.

Zmienne ustawiane przez serwer WWW:

HTTP_ACCEPT_LANGUAGE

Zawartość nagłówka „Accept-Language” z aktualnego zapytania, jeśli taki istnieje, np. ‚en’.

HTTP_CONNECTION

Zawartość nagłówka „Connection” z aktualnego zapytania, jeśli taki istnieje, np. ‚Keep-Alive’.

HTTP_HOST

Zawartość nagłówka „Host” z aktualnego zapytania, jeśli taki istnieje.

HTTP_REFERER

Adres strony (jeśli taka była), która wskazała przeglądarkę do tej strony. Wartość ta jest ustawiana przez przeglądarkę – nie wszystkie to robią.

Zmienne ustawiane przez serwer WWW:

HTTP_USER_AGENT

Zawartość nagłówka „User-Agent” z zapytania, jeśli taki istnieje. Jest to ciąg informujący o przeglądarce która została użyta do obejrzenia bieżącej strony, np. Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586). Można użyć funkcji get_browser() aby dopasować funkcjonalność strony do przeglądarki użytkownika.

REMOTE_ADDR

Adres IP z którego użytkownik połączył się z serwerem.

REMOTE_PORT

Port używany do komunikacji pomiędzy użytkownikiem a serwerem.

SCRIPT_FILENAME

Ścieżka do aktualnie wykonywanego skryptu.

Zmienne ustawiane przez serwer WWW:

SERVER_ADMIN

Wartość podana dla opcji SERVER_ADMIN w konfiguracji serwera WWW. Jeśli skrypt działa na wirtualnym serwerze, to będzie to wartość podana dla tego wirtualnego serwera.

SERVER_PORT

Port na serwerze którego użyto do połączenia. Dla normalnych połączeń będzie to ’80’.

SERVER_SIGNATURE

Ciąg zawierający wersję i nazwę wirtualnego hosta który jest dodawany do stron generowanych przez serwer.

Zmienne ustawiane przez serwer WWW:

SCRIPT_NAME

Zawiera ścieżkę do aktualnie wykonywanego pliku. Jest to przydatne do skyptów, które muszą wskazywać samego siebie.

REQUEST_URI

URI który został podany aby uzyskać dostęp do tej strony.

Zmienne ustawiane przez PHP:

argv

Tablica argumentów przkazywanych do skryptu. Jeśli skrypt jest uruchamiany z linii poleceń, to zmienna ta daję dostęp do argumentów w stylu języka C. Jeśli jest wywołany przez metodę GET, to zmienna ta zawierać będzie ciąg parametrów (query string).

argc

Zawiera liczbę parametrów podanych podanych do skryptu w linii poleceń (jeśli skrypt został wywołany z linii poleceń).

PHP_SELF

Nazwa pliku aktualnie wykonywanego skryptu, względna do katalogu głównego dokumentów. Ta zmienna jest niedostępna jeśli PHP jest uruchamiany z linii poleceń.

Zmienne ustawiane przez PHP:

HTTP_COOKIE_VARS

Tablica asocjacjna zmiennych przekazanych do skryptu przez HTTP cookies. Dostępna tylko jeśli włączone zostało śledzenie zmiennych przez ustawienie w konfiguracji PHP opcji track_vars lub komendą <?php_track_vars?>.

HTTP_GET_VARS

Tablica asocjacjna zmiennych przekazanych do skryptu przez metodę GET. Dostępna tylko jeśli włączone zostało śledzenie zmiennych przez ustawienie w konfiguracji PHP opcji track_vars lub komendą <?php_track_vars?>.

HTTP_POST_VARS

Tablica asocjacjna zmiennych przekazanych do skryptu przez metodę POST. Dostępna tylko jeśli włączone zostało śledzenie zmiennych przez ustawienie w konfiguracji PHP opcji track_vars lub komendą <?php_track_vars?>.

Stałe

W PHP występują także tzw. stałe, czyli identyfikatory znakowe, których wartości nie można zmienić. Stałych, w odróżnieniu od zmiennych, używa się bez znaku dolara na początku. W PHP występuje kilka zmiennych ustawianych przez parser.

Stałe ustawiane przez PHP:

__FILE__

Nazwa pliku ze skryptem który jest aktualnie przetwarzany. Jeśli stała ta użyta jest wewnątrz pliku który został zainkludowany (o poleceniu include w dalszej części kursu), to podana zostanie nazwa pliku zainkludowanego, a nie pliku nadrzędnego.

Stałe ustawiane przez PHP:

__LINE__

Numer linii w skrypcie która aktualnie jest przetwarzana. Jeśli stała ta użyta jest wewnątrz pliku który został zainkludowany, to podany zostanie numer linii przetwarzanej w pliku zainkludowanym.

PHP_VERSION

Ciąg reprezentujący wersję parsera PHP aktualnie używaną.

PHP_OS

Nazwa systemu operacyjnego na którym uruchamiany jest parser PHP.

TRUE

Logiczna wartość prawdy.

FALSE

Logiczna wartość fałszu.

Stałe

Stałe mogą być definiowane przez użytkownika za pomocą funkcji define(), która przyjmuje 2 parametry: nazwę stałej i wartość do niej przypisaną.

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

OPERATORY

Co to jest?

Operatory są to najprościej mówiąc symbole, które służą do operacji na zmiennych. Operatory dzielą się na operatory arytmetyczne, które służą do operacji na liczbach, operatory przypisania służące do przypisywania zmiennym wartości, operatory operacji bitowych, operatory porównania niezbędne do instrukcji warunkowych, operator kontroli błędów, operator wykonania służący do uruchamiania zewnętrznych programów, operatory inkrementacji i dekrementacji, operatory logiczne i operatory ciągu.

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.

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.

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

UWAGA:

Operator AND daje TRUE tylko gdy mamy do czynienia z samą prawdą:

TRUE and TRUE ➔ TRUE

Ale

TRUE and FALSE == FALSE and TRUE == FALSE and FALSE ➔ FALSE

Operator OR daję FALSE tylko gdy mamy do czynienia z samym fałszem:

FALSE or FALSE ➔ FALSE

Ale

TRUE or FALSE == FALSE or TRUE == TRUE or TRUE ➔ TRUE

UWAGA: Operatory logiczne AND i OR są jak MNOŻENIE i DODAWANIE w matematyce – mają swojepriorytety (AND jest pierwszy w kolejności do obliczeń niż OR):

MATEMATYKA: 2+3*3 = 2+9 = 11 ale (2+3)*3 = 5*3 = 15

Podobnie jak w matematyce tak i w logice można zmienić priorytet wykonywania operacji za pomocą nawiasów…

Operatory porównania i logiczne – Przykłady

$a = 10; $b = 5;

$a > $b => TRUE

$a <= $b => FALSE

($a – 5) <= $b => TRUE

$b <= $a => TRUE

$a != $b && $a > $b => TRUE

$a == $b && $a > $b => FALSE

$a == $b || $a > $b => TRUE

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.

Operatory inkrementacji i dekrementacji

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 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)

STRUKTURY KONTROLNE

Instrukcje warunkowe

Instrukcje warunkowe są podstawą każdego języka programowania. Używa się jej do wykonania pewnej instrukcji (lub bloku instrukcji), ale tylko w pewnych okolicznościach – zostanie spełniony określony warunek (lub cały zestaw warunków).

Składnia instrukcji warunkowej

```
<?php
    if(wyrażenie_warunkowe)
      instrukcja wykonywana jeśli spełniony zostanie warunek
    elseif(inne_wyrażenie_warunkowe)
      instrukcja wykonywana jeśli spełniony zostanie drugi warunek, a pierwszy nie
    else
      instrukcja wykonywana jeśli nie zostanie spełniony żaden z warunków
?>
```

Wyrażeniem warunkowym jest w zasadzie dowolne wyrażenie, ponieważ za warunek uznawane jest wszystko co zwraca wartość, czyli wszystkie zmienne, wyrażenia logiczne, funkcje itp. Za spełniony warunek uznawana jest wartość większa od zera.

Przykład instrukcji warunkowej:

```
<?php
        $a = 2;
        $b = 5;
        $c = 1;
        if($a > $b)
          echo "$a jest większe od $b";
        elseif ($b > $c)
          echo "$b jest większe od $c";
        else
          echo "$c jest większe od $a i $b";
        if($a)
          echo "Zmienna $a ma wartość większą od zera";
?>
```

Instrukcje warunkowe

Jeśli chcemy, aby po sprawdzeniu warunku wykonane zostało nie jedno, ale kilka poleceń, to te polecenia trzeba ująć w nawiasy klamrowe. Bez tego warunkiem objęta by była tylko jedna instrukcja po instrukcji if.

```
<?php
        $a = 2;
        $b = 5;
        $w = 0;
        if($a  == 0)
          echo "$a jest równe zero – nie dzielimy przez zero";
        else
         {       $w = $b/$a;
                 echo "wynik dzielenia B przez A to $w";     }
?>
```

Zagnieżdżanie instrukcji warunkowych

Instrukcje mogą być zagnieżdżane wewnątrz siebie.

```
<?php
    $a = 6;
    $b = 5;
    $c = 1;
    if($a > $b){
        echo "$a jest większe od $b";
        if($a > $c)
            echo " i od $c"; // Powinien zostać wyświetlony napis "6 jest większe
        od 5 i od 1"
    }
?>
```

OR i AND

Oczywiście możliwe jest korzystanie z warunków bardziej złożonych niż pojedyncze porównanie wielkości zmiennych – do łączenia warunków niezbędne jest wykorzystanie operatorów logicznych opisanych w poprzednim rozdziale.

Operator logiczny OR (lub) ma niższy priorytet niż operator AND (i), więc aby sprawdzić jakiś warunek gdzie konieczna jest inna kolejność, niezbędne jest użycie nawiasów grupujących warunki.

OR i AND - grupowanie

Na przykład: chcemy aby jakaś instrukcja była wykonana jeśli zmienna $a jest większa od $b lub $c, i zmienna $d była równa $e.

Jeśli chcielibyśmy zapisać to bez żadnych nawiasów:

```
$a > $b || $a > $c && $d == $e
```

to efekt byłby zupełnie inny od zamierzonego: instrukcja była by wykonana jeśli $a było by większe od $b, lub jeśli $a było by większe od $a i $d było by równe $e.

Poprawna konstrukcja to `($a > $b || $a > $c) && $d == $e`.

Pętla FOR

Czasem zachodzi potrzeba wykonania jakiejś czynności określoną ilość razy.

Z pomocą przychodzi jedna z najczęściej używanych składni w większości języków programowania, czyli pętla FOR.

Ogólny zapis FOR:

```
<?php
    for( inicjalizacja zmiennych ; sprawdzenie warunku ; modyfikacja zmiennych)
     {          blok wyrażeń          }
?>
```

Pętla FOR

W pętli podaje się 3 wyrażenia jako parametry: inicjalizację zmiennych, czyli ustawienie początkowych wartości dla zmiennych kontrolujących pętlę, sprawdzenie warunku, czyli wyrażenie logiczne kontrolujące pętlę – pętla będzie wykonywana dopóki ten warunek jest prawdziwy, oraz modyfikację zmiennych kontrolujących pętlę – bez tego pętla będzie wykonywała się w nieskończoność (oczywiście wartość tych zmiennych można modyfikować wewnątrz pętli, ale jest to niezalecane).

Przykład najprostszej pętli, która wypisze cyfry od 1 do 10:

```
<?php
    for( $x = 1; $x <= 10; $x++ )
        echo $x."<br>";
?>
```

Pętla while

Innym rodzajem pętli jest pętla WHILE. Jest ona wykorzystywana w sytuacjach, kiedy niezbędne jest wykonywanie jakiejś operacji dopóki nie zostanie spełniony warunek.

```
<?php
    while( warunek ){
            ...
        instrukcje
            ...
    }
?>
```

Pętla do…while

Odmianą pętli while jest pętla do…while. Od zwykłej pętli while różni się ona tym, że polecenia zawarte w pętli **będą przynajmniej raz wykonane** – w przypadku pętli while tak być nie musi, to znaczy jeśli za pierwszym razem warunek nie zostanie spełniony to polecenia z pętli nigdy nie zostaną wykonane. W przypadku tej pętli zostano one wykonane przynajmniej ten pierwszy raz.

```
<?php
    do {
        ...
        instrukcje
        ...
        } while( warunek );
?>
```

Przykłady pętli:

Wyświetlanieliczb parzystych (mniejszych i równych 100)

```
<?php
    for($x = 2; $x <= 100; $x+=2) {
        echo $x." ";
    }
?>
```

Zsumowanie liczb parzystych (mniejszych i równych 100)

```
<?php
    $s = 0;
    for($x = 2; $x <= 100; $x+=2) {
        $s = $s + $x;
    }
?>
```

Składnia switch

Składnia switch jest instrukcją warunkową, ale jedną zmienną można porównać nie z jedną wartością, ale z kilkoma. Niestety nie można konstruować złożonych warunków – możliwe jest tylko proste porównywanie (równoważne instrukcji: if($zmienna=="wartość") instrukcja).

Składnia switch

```
<?php
switch($zmienna){
  case 'wartość1':
      ...
      instrukcje
      ...
      break;
  case 'wartość2':
    ...
      instrukcje
      ...
      break;
  default:
    ...
      instrukcje
      ...
}
?>
```

Przykład użycia składni switch

```
<?php
    $i = 3;
    switch($i){
        case 0:
        case 1:
        case 2:
        case 3:
            echo "Zmienna $i jest mniejsza bądź równa od trzech\n";
            break;
        case 4:
            echo "Zmienna $i jest równa cztery\n";
            break;
        default:
            echo "Zmienna $i jest większa od czterech\n";
    }
?>
```

KONIEC cz.2