przydatne strony

http://home.agh.edu.pl/~wojnicki/wiki/_media/pl:paw:dphp0.pdf

http://php-mysql-mvc.gajdaw.pl/instalacja/index.htm

http://apache.org/

http://php.net/manual/pl/

http://webinside.pl/

http://www.w3schools.com/php/php_if_else.asp

Programowanie w PHP- *potrzebne narzędzia*

1. **serwer HTTP (zwany także serwerem WWW) z obsługą PHP – np. Apache**
2. **aplikacja zwana interpreterem PHP**

PHP jest wykonywany po stronie serwera. Oznacza to, że PHP nie jest interpretowany (przetwarzany) przez przeglądarkę, lecz przez specjalny program na serwerze.

3. **serwer bazodanowy MySQL**

Implementacja PHP wraz z serwerem WWW Apache oraz serwerem baz danych MySQL określana jest jako platforma AMP (w środowisku Linux – LAMP, w Windows – WAMP).

Te trzy komponenty można zainstalować własnoręcznie lub skorzystać z gotowych pakietów.

Downloading Apache for Windows

The Apache HTTP Server Project itself does not provide binary releases of software, only source code. Individual committers *may* provide binary packages as a convenience, but it is not a release deliverable.

If you cannot compile the Apache HTTP Server yourself, you can obtain a binary package from numerous binary distributions available on the Internet.

Popular options for deploying Apache httpd, and, optionally, PHP and MySQL, on Microsoft Windows, include:

- ApacheHaus
- Apache Lounge
- BitNami WAMP Stack
- WampServer
- XAMPP

Wymaga VISUALSTUDIO

Do nauki programowania w PHP wykorzystamy XAMPP

PHP

PHP jest językiem skryptowym działającym po stronie serwera. Osadza się go w kodzie HTML w postaci bloków ograniczonych znacznikami <?php ?>, które są przekształcane na HTML podczas każdorazowego odświeżenia strony.

Kod PHP jest wykonywany po stronie serwera, który interpretuje składnie i wysyła odpowiednio zmodyfikowany kod HTML.

Użytkownik strony może zobaczyć jedynie efekt, nie mając wglądu do napisanego przez nas skryptu.

Język PHP stworzony został w 1994 roku przez Rasmusa Lerdorfa. Jest to produkt Open Source, czyli każdy ma swobodny dostęp do jego kodu źródłowego, który można dowolnie modyfikować i rozprowadzać. Strona główna PHP wraz ze szczegółową specyfikacją to www.php.net.

Jak PHP współpracuje ze stroną WWW?

PHP jest językiem server-side, tj. pracuje po stronie serwera WWW. Przeciwieństwem są języki client-side pracujące po stronie przeglądarki użytkownika (np. JavaScript w wersji wykonywanej po stronie klienta). Aby wykorzystać PHP na własnej stronie, należy upewnić się, że serwer WWW ma zainstalowaną jego obsługę. W jaki sposób PHP generuje dynamiczne strony WWW?

Kiedy wpisujemy adres w przeglądarce internetowej, żądanie wyświetlenia strony kierowane jest do serwera HTTP Jeśli stwierdzi on, na podstawie rozszerzenia pliku, że dany dokument zawiera kod PHP, kieruje do interpretera żądanie przetworzenia podanego pliku. Interpreter wyszukuje w jego treści znaczniki PHP wplecione w statyczny kod HTML i zastępuje je wynikiem ich przetworzenia. Utworzony kod HTML jest zwracany serwerowi, a

ten wysyła go do przeglądarki.

Konfiguracja PHP-plik php.ini

Dyrektywa	Domyślna Wartość	Do czego służy?
allow_url_fopen	On	Od niej zależy możliwość używania funkcji: • file_get_contents • fopen
asp_tags	Off	Możliwość użycia tagów <% %> zamiast <? php ?>.
display_errors	On	Odpowiada za wyświetlanie błędów użytkownikom lub ukrywanie ich.
error_log	no value	Określa, gdzie będą zapisywane logi. Podajemy ścieżkę od katalogu ROOT do pliku z logami np. "/logs/php_errors.log".
error_reporting	Off	Decyduje, jaki rodzaj błędów będzie logowany: • E_ALL • E_NOTICE • E_STRICT Stosowane kombinacje: E_ALL & ~E_NOTICE

Konfiguracja PHP-plik php.ini

include_path	.:/:/usr/local/php/pear5	Wskazuje gdzie funkcje require(), include(), fopen(), file(), readfile() and file_get_contents() mają szukać plików.
log_errors	Off	Ustala, czy błędy będą logowane (dotyczy wyłącznie logowania w zakresie Klienta, logi systemowe gromadzone są przez home.pl w ramach fizycznego serwera).
magic_quotes_gpc	On	W przeszłości wykorzystywane głównie do zabezpieczenia przed SQL Injection. Kliknij tutaj, aby sprawdzć jak wyłączyć funkcję magic_quotes_gpc na serwerze w home.pl.
memory_limit	128m	Określa limit pamięci dla wykonywanych skryptów na serwerze. Domyślnie ustawiona wartość to 128M. Użytkownicy korzystający z serwerów Linii profesjonalnej mogą zwiększyć ten parametr do wartości 256M.
post_max_size	64M	Wielkość danych przesyłanych metodą POST.
register_globals	Off	Zmienne globalne odpowiadają za możliwość korzystania m.in. ze zmiennych przesyłanych do serwera.

Czym jest język PHP

PHP (angielski akronim rekurencyjny, którego rozwinięcie to PHP Hypertext Preprocessor), pierwotnie nazwany Personal Home Page
- skryptowy język programowania, służący przede wszystkim do tworzenia dynamicznych stron WWW i wykonywany

w tym przypadku po stronie serwera, z mozliwością zagnieżdżania w HTML (bądź w XHTML)

Udostępniany jest na zasadach licencji open- source. Jego składnia bazuje na językach C, Java i Perl.

Czym jest język PHP

PHP jest językiem skryptowym działającym po stronie serwera.

Osadza się go w kodzie HTML w postaci bloków ograniczonych znacznikami <?php ?>, które są przekształcane na HTML podczas każdorazowego odświeżenia strony.

Kod PHP jest wykonywany po stronie serwera, który interpretuje składnie i wysyła odpowiednio zmodyfikowany kod HTML. Użytkownik strony może zobaczyć jedynie efekt, nie mając wglądu do napisanego przez nas skryptu.

Mamy cztery różne pary otwierających i zamykających znaczników, które mogą być użyte w php:

- <?php ?> i <script language="php"> </script> - są dostępne zawsze.
- <? ?> i <% %> czyli krótkie znaczniki i znaczniki w stylu ASP, mogą być włączane i wyłączane w pliku konfiguracyjnym php.ini. Może są one wygodne, jednakże są one mniej przenośne, i zasadniczo nie polecane.

Instrukcje wyjścia

W języku PHP instrukcje:

- echo
- print

pozwalają na wypisanie na stronie www wyników pracy programu

Z punktu widzenia składni języka PHP „echo” oraz „print” nie są ani funkcjami ani instrukcjami. Jest to coś pośredniego nazywanego w dokumentacji „konstrukcją językową”.

To znaczy, że możemy używać „echo” oraz „print” zarówno tak, jakby to były funkcje (czyli otaczając parametr nawiasami) oraz rezygnując z nawiasów. Wszystkie cztery poniższe instrukcje są poprawne:

Czym jest język PHP

PHP (angielski akronim rekurencyjny, którego rozwinięcie to PHP Hypertext Preprocessor), pierwotnie nazwany Personal Home Page
- skryptowy język programowania, służący przede wszystkim do tworzenia dynamicznych stron WWW i wykonywany
w tym przypadku po stronie serwera, z mozliwością zagnieżdżania w HTML (bądź w XHTML)

Udostępniany jest na zasadach licencji open- source. Jego składnia bazuje na językach C, Java i Perl.

Wszystkie cztery poniższe instrukcje są poprawne:

```
echo 'wiosna';
 echo('lato');
print 'jesień';
print('zima');
```

W każdym przypadku można wpisać cudzysłów "" zamiast apostrofu ' ', ale zaleca się używać apostrofu ponieważ tekst w apostrofach jest nieco szybciej "czytany" przez interpreter PHP

Instrukcje

wiosna
lato
jesień
zima

```
    <meta charset="UTF-8" />
</head>
<body>
<h1>Instrukcje wyjścia. Pierwszy program</h1>
<?php
echo 'wiosna <br>';
echo('lato  <br>');
print 'jesień  <br>';
print('zima');
?>
</body>
</html>
```

W instrukcji echo mamy możliwość podania wielu argumentów oddzielonych przecinkami

np.:

echo 'jeden', 'dwa', 'trzy'; - jest to jednak sposób niezalecany

W instrukcji print możemy podać tylko jeden argument:

Print 'jeden dwa trzy';

PHP ustala typ danych na podstawie kontekstu

Nazwa zmiennej w PHP :

- **zaczyna się od znaku dolara $**
- **pierwszy znak może być znakiem podkreślenia _ lub dowolną literą**
- **pozostałe znaki mogą być cyframi, literami lub znakiem podkreślenia**
- **rozróżnialne są małe i duże litery.**

wyświetlanie danych zawartych w cudzysłowach przebiega wolniej niż w apostrofach

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>Instrukcje wyjścia.  Zmienne </title>
    <meta charset="UTF-8" />
</head>
<body>
<h1>Instrukcje wyjścia.  Zmienne </h1>
<?php
$a=5;
$b="cudzyslow ";//może być cudzysłów
$c=' apostrof'; // moze być apostrof
echo "wiosna $a<br>";
echo  'lato $b <br>';
print  "zima  $c <br>";
print 'jesień $c <br>';
?>
</body>
</html>
```

Cudzysłowy zezwalają na "proste" umieszczanie wewnątrz tekstu wartości zmiennych

wiosna 5
lato $b
zima apostrof
jesień $c

wstawianie zmiennych w ten sposób jest kilka razy wolniejsze, niż łączenie ich z ciągiem operatorem kropki.

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>Instrukcje wyjścia.  Zmienne </title>
    <meta charset="UTF-8" />
</head>
<body>
<h1>Instrukcje wyjścia.  Zmienne </h1>
<?php
$a=5;
$b="cudzyslow ";//może być cudzysłów
$c=' apostrof'; // moze być apostrof
echo "wiosna". $a."<br>";
echo  'lato'. $b.' <br>';
print  "zima".$c." <br>";
print 'jesień '.$c.' <br>';
?>
</body>
</html>
```

```
echo 'Ala';
echo('Ala');
print 'Ala';
print('Ala');
```

1. **Z czterech sposobów użycia instrukcji echo oraz print wybieramy jeden i stosujemy go konsekwentnie. Nie należy łączyć kilku z podanych metod.**

2. **Pomiędzy echo oraz print nie ma żadnej istotnej różnicy. Należy jednak zdecydować się i stosować wyłącznie jedną lub drugą z nich.**

3. **Podobnie, należy zdecydować się na stosowanie nawiasów, bądź rezygnację z nich.**

UWAGA: Instrukcja phpinfo();
spowoduje wyświetlenie parametrów PHP

test.php

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>test  </title>
    <meta charset="UTF-8" />
</head>
<body>
<?php
echo 'to jest test instrukcji echo';
?>
</body>
</html>
```

po wykonaniu powyższego skryptu przez maszynę PHP otrzymamy kod HTML:

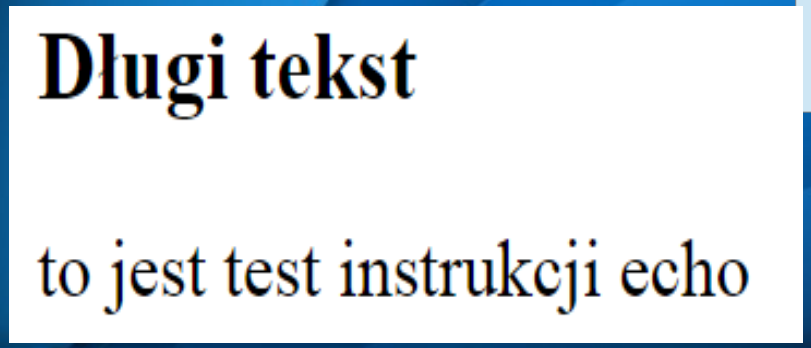
```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>test  </title>
    <meta charset="UTF-8" />
</head>
<body>
to jest test instrukcji echo
</body>
</html>
```

1. uruchamiamy przeglądarkę internetową;
2. wprowadzamy adres skryptu, np. http://localhost/test.php;
3. przeglądarka łączy się z serwerem (tj. procesem apache) uruchomionym na komputerze lokalnym (tj. na tym samym, na którym jest uruchomiona przeglądarka);
4. przeglądarka wysyła do serwera zapytanie HTTP zawierające nazwę pliku test.php
5. serwer apache odbiera zapytanie HTTP;
6. po odebraniu zapytania serwer apache odczytuje plik test.php;
7. na podstawie rozszerzenia nazwy pliku (.php), apache „wie”, że ma przetworzyć plik procesorem PHP;
8. serwer apache uruchamia procesor PHP i przekazuje mu zawartość pliku test.php;
9. procesor PHP odnajduje fragment otoczony znacznikami <?php oraz ?>; fragment ten jest wycięty z dokumentu oryginalnego (tj. test.php) i poddany wykonaniu przez maszynę PHP;
10. po wykonaniu wyciętego skryptu, maszyna PHP otrzymuje pewien tekst, wydrukowany przez skrypt (w przykładzie tekstem tym jest 'to jest test instrukcji echo');

Etapy przetwarzania pliku php:

11. tekst otrzymany po wykonaniu skryptu jest wklejany do orginalnego dokumentu, w miejsce, z którego wycięto skrypt otoczony znacznikami <?php i ?>;
12. tak otrzymany dokument maszyna PHP przekazuje do serwera apache;
13. serwer apache odbiera stronę wygenerowaną przez maszynę PHP i wysyła ją do przeglądarki;
14. przeglądarka wyświetla otrzymany kod HTML na ekranie.

W skrócie, interpretator języka PHP zastąpi kod skryptu PHP otoczony znacznikami <?php oraz ?> tekstem, jaki jest drukowany przez skrypt.



Dostępne są funkcje, które mogą sprawdzic typ

- **danych: is_array()**
- **is_int()**
- **is_float()**

... itp.

oraz istnienie samej zmiennej i jej

- **wartosci: isset()**
- **is_null()**
- **empty()**

Typy danych w PHP

PHP obsługuje następujące typy proste:

- **boolean:**
  True/False
- **$a = True;**
- **liczba całkowita (integer)**

może być zapisana w notacji dziesiętnej, szesnastkowej (0x) lub ósemkowej (0),

$a=0x6F44;

$b=07647

- **liczba zmiennoprzecinkowa (ang. floating point numbers albo skrótowo float)**

$a = 1.234; $a = 1.2e3; $a = 7E-10;

Value of variable ($var)	isset($var)	empty($var)	is_null($var)
"" (an empty string)	bool(true)	bool(true)	bool(false)
" " (space)	bool(true)	bool(false)	bool(false)
FALSE	bool(true)	bool(true)	bool(false)
TRUE	bool(true)	bool(false)	bool(false)
array() (an empty array)	bool(true)	bool(true)	bool(false)
NULL	bool(false)	bool(true)	bool(true)
"0" (0 as a string)	bool(true)	bool(true)	bool(false)
0 (0 as an integer)	bool(true)	bool(true)	bool(false)
0.0 (0 as a float)	bool(true)	bool(true)	bool(false)
var $var; (a variable declared, but without a value)	bool(false)	bool(true)	bool(true)
NULL byte ("\ 0")	bool(true)	bool(false)	bool(false)

Zmienną usuwamy poleceniem unset($zmienna).

Typy danych w PHP

- **łańcuch znaków (string),**

  echo ’przykładowy tekst’;

- **tablica (array)**

  $tablica=array(1,2,3,’cos’,5);

  echo $tablica[0];

- **obiekt (object)**

- **identyfikator zasobów (resource) jest specjalną zmienną, przechowującą odnośnik do zewnętrznego źródła zasobów**

Typy danych w PHP

- **łańcuch znaków (string),**

  echo ’przykładowy tekst’;
- **tablica (array)**

  $tablica=array(1,2,3,’cos’,5);

  echo $tablica[0];
- **obiekt (object)**
- **identyfikator zasobów**

  (resource) jest specjalną zmienną, przechowującą odnośnik do zewnętrznego źródła zasobów

Konwersja (rzutowanie) typów

PHP potrafi sam rozpoznać typ informacji przypisanej do zmiennej oraz automatycznie konwertować go w zależności od potrzeb. Przykładowo liczby ułamkowe użyte tam, gdzie potrzeba całkowitych, są zaokrąglane w górę do liczby całkowitej. Wartości logiczne mogą być reprezentowane cyframi 0 (FALSE) oraz 1 (TRUE). Ciągi tekstowe mogą być konwertowane do liczb, jeżeli pierwszy znak (pomijając wiodące białe znaki) jest cyfrą. W przeciwnym przypadku PHP dobiera wartość 0.

Możemy sami wymusić konwersję typów:

```php
<?php
// wyświetl liczbę całkowitą jako ułamek
echo (float) 10;
```

W nawiasie przed wartością piszemy angielską nazwę typu: integer, int, string, boolean, float, double. Nie należy stosować jakiejkolwiek konwersji typów złożonych: tablic, obiektów, zasobów, gdyż w każdym z tych przypadków informacje zostają całkowicie utracone; zamiast nich zwracana jest nazwa typu złożonego

Inny przykład umieszczania kodu *PHP w HTML*

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>kod PHP w HTML</title>
    <meta charset="UTF-8" />
</head>
<body >
<h1>HTML poprzecinany kodem PHP-łatwo o pomyłkę</h1>
    <?php
    $wyrazenie_logiczne=true;
if ($wyrazenie_logiczne==true) {
?>
<strong>prawda </strong>
<?php
} else {
?>
<strong>fałsz </strong>
<?php
}
?>
</body>
</html>
```

gettype()

PHP posiada funkcję gettype($zmienna) zwracającą nazwę aktualnego typu danych zawartych w zmiennej:

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>Instrukcje wyjścia.  Zmienne </title>
    <meta charset="UTF-8" />
</head>
<body>
<h3>Funkcja gettype() </h3>
<?php
$a=5;
echo 'zmienna jest typu '.gettype($a).'<br>';
$b='jakis tekst';
echo 'zmienna jest typu '.gettype($b).'<br>';
unset($b);
echo 'zmienna jest typu '.gettype($b);
?>
</body>
</html>
```

Funkcja gettype()

zmienna jest typu integer
zmienna jest typu string
zmienna jest typu NULL

Definiowanie stałej

Stała to obszar w pamięci, przechowujący pewną wartość, która jednak nie może ulec zmianie podczas wykonywania skryptu.

define(„nazwa_stalej”, „wartosc”)

```
<body>
<h1>Definiowanie stałej </h1>
<?php
        define ("KWOTA_DO_ODJECIA", 436.20);
        define ("STAWKA_PODATKOWA", 0.19);  // czyli 19%
        echo 'Podatek od dochodu 5000 PLN  wynosi: ';
        echo  5000 * STAWKA_PODATKOWA - KWOTA_DO_ODJECIA;
       echo ' PLN';
    ?>

</body>
```

OPERATORY

ARYTMETYCZNE

Operator	Nazwa	Składnia	Opis
/	Dzielenie	wyrażenie / wyrażenie	Reprezentuje wynik dzielenia. Drugie wyrażenie nie może być zerem.
%	Dzielenie modulo	wyrażenie % wyrażenie	Reprezentuje resztę z dzielenia. Drugie wyrażenie nie może być zerem.
*	Mnożenie	wyrażenie * wyrażenie	Reprezentuje iloczyn dwóch wyrażeń.
+	Dodawanie	wyrażenie + wyrażenie	Reprezentuje sumę dwóch wyrażeń.
-	Odejmowanie	wyrażenie - wyrażenie	Reprezentuje różnicę dwóch wyrażeń.
.	Łączenie (Konkatenacja)	wyrażenie . wyrażenie	Reprezentuje połączenie dwóch wyrażeń w ciąg tekstowy.
++	Postinkrementacja (zwiększenie)	**$zmienna++**	Reprezentuje wartość zmiennej, a następnie zwiększa ją o 1.
++	Preinkrementacja (zwiększenie)	**++$zmienna**	Zwiększa wartość zmiennej o 1, a następnie reprezentuje ją.
--	Postdekrementacja (zmniejszenie)	**$zmienna--**	Reprezentuje wartość zmiennej, a następnie zmniejsza ją o 1.
--	Predekrementacja (zmniejszenie)	**--$zmienna**	Zmniejsza wartość zmiennej o 1, a następnie reprezentuje ją.

OPERATORY

PRZYPISANIA

Operator	Składnia	Równoważna postać	Opis
/=	$zmienna /= wyrazenie	$zmienna = $zmienna / wyrażenie	Dzieli zmienną przez wyrażenie i umieszcza w niej wynik. Wyrażenie nie może być zerem.
%=	$zmienna %= wyrazenie	$zmienna = $zmienna % wyrażenie	Umieszcza w zmiennej resztę z dzielenia tej zmiennej przez wyrażenie, które oczywiście nie może być zerem.
*=	$zmienna *= wyrazenie	$zmienna = $zmienna * wyrażenie	Mnoży zmienną przez wyrażenie i zapisuje w niej wynik.
+=	$zmienna += wyrazenie	$zmienna = $zmienna + wyrażenie	Dodaje do zmiennej wyrażenie i zapisuje w niej wynik.
-=	$zmienna -= wyrazenie	$zmienna = $zmienna - wyrażenie	Odejmuje od zmiennej wyrażenie i zapisuje w niej wynik.
.=	$zmienna .= wyrazenie	$zmienna = $zmienna . wyrażenie	Dołącza do zmiennej tekstowej nowy fragment.

OPERATORY

RELACJI

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y

OPERATORY

LOGICZNE

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

Zastosowanie OPERATORA *Kropka*

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>Długi tekst </title>
    <meta charset="UTF-8" />
</head>
<body>
<h3>Długi tekst </h3>
<?php
$tekst = 'wiersz 1<br>';
$tekst .= 'wiersz 2<br>';
$tekst .= 'wiersz 3<br>';
$tekst .= 'wiersz 4<br>';
echo $tekst;
?>
</body>
</html>
```

Długi tekst

wiersz 1
wiersz 2
wiersz 3
wiersz 4

Zmienne i operatory

```
<body >
<h3>poniżej wynik działania skryptu php</h3>

    <?php

    // W zmiennych zapamiętamy boki prostokąta, a
       // następnie obliczymy pole i obwód kwadratu.
       $bokA = 5;
       $bokB = 7;
       echo  'Pole prostokąta o bokach '. $bokA .' i '. $bokB. ' wynosi: '.$bokA*$bokB.
             " a obwód: " . 2*($bokA+$bokB) ;
    ?>
</body>
```

poniżej wynik działania skryptu php

Pole prostokąta o bokach 5 i 7 wynosi: 35 a obwód: 24

Instrukcja warunkowa

```
if(wyrażenie_warunkowe)
  instrukcja wykonywana jeśli spełniony zostanie warunek;
else
      instrukcja wykonywana jeśli nie jest spełniony
    warunek;
  if(wyrażenie_warunkowe) {
  instrukcje wykonywane jeśli spełniony zostanie warunek;
  }
  else {
  instrukcje wykonywane jeśli nie jest spełniony warunek;
  }
```

```
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>Instrukcja warunkowa </title>
    <meta charset="UTF-8" />
</head>
<body>
<h1>Instrukcja warunkowa </h1>
<?php
$liczba=4;
      if ($liczba == 3)
    {
    echo 'Podana liczba to 3';
    }
else
    {
    echo 'Podano inną liczbę niż 3';
    }
    ?>
</body>
</html>
```

Instrukcja warunkowa

```
if(wyrażenie_warunkowe) {
   instrukcje wykonywane jeśli spełniony zostanie warunek; }
 elseif(inne_wyrażenie_warunkowe) {
   instrukcje wykonywane jeśli spełniony zostanie drugi warunek,
a  pierwszy nie ;}
   else {
   instrukcje wykonywane jeśli nie zostanie spełniony żaden
z  warunków; }
```

```php
<body>
<h1>Instrukcja warunkowa </h1>
<?php
$liczba=4;
      if ($liczba == 0)
    {
    echo 'Podana liczba jest równa 0';
    }
else if ($liczba<0)// mozna tez napisać  elseif
    {
    echo 'Podano liczbę ujemną';
    }
    else {
    echo 'Podano liczbę dodatnią';
    }
    ?>
</body>
```

Wyrażenie warunkowe-skrócona *instrukcja if*

```
warunek ? instrukcja_true : instrukcja_false
```

```
if (warunek) {
instrukcja_true;
} else {
instrukcja_false;
}
```

Wyrażenie warunkowe-skrócona *instrukcja if*

```
<body>
<h3>wyrazenie warunkowe</h3>
<?php
$a=3;
$b=8;
echo ($a>$b ?  "Liczba $a jest większa od liczby $b" : "Liczba $a nie jest większa od liczby $b");
    ?>
    <h3>taki sam wynik uzyskany instrukcją if</h3>
    <?php
$a=3;
$b=8;
if ($a>$b) {
print("Liczba $a jest większa od liczby $b");
} else {
print("Liczba $a nie jest większa od liczby $b");
}
    ?>
</body>
```

wyrazenie warunkowe

Liczba 3 nie jest większa od liczby 8

taki sam wynik uzyskany instrukcją if

Liczba 3 nie jest większa od liczby 8

Instrukcja switch – instrukcja *wyboru*

```
switch (wyrażenie) {
 case wartosc1: dzialanie1;
                break;
 case wartosc2: dzialanie2;
                break;
        ….
        default: dzialanie;
 }
```

instrukcja switch

Wartość zmiennej a to 2

```
<body>
<h3>instrukcja switch</h3>
<?php
   $a = 2;
switch ($a) // sprawdzamy zmienną $a
{
case 1:
  echo "Wartość zmiennej a to 1";
  break;

case 2:
  echo "Wartość zmiennej a to 2";
  break;

case 3:
  echo "Wartość zmiennej a to 3";
  break;

default:
  echo "Żadna z powyższych";
  break;
}
    ?>
</body>
```