1. Wstawianie skryptów PHP

<?php ?>
<? ?>
<SRIPT LANGUAGE=”PHP”> </SCRIPT>

2. Wstawianie komentarzy

/* */ Komentarz może zajmować wiele wierszy
// Po dwóch ukośnikach wszystko do końca wiersza jest komentarzem
Po znaku „hash” wszystko do końca wiersza jest komentarzem

3. Przesyłanie danych do przeglądarki internetowej (funkcje wypisujące tekst)

print – funkcja wypisująca podaną wartość
echo – funkcja wypisująca podaną wartość (podobna do print)
printf – funkcja pozwalająca na wypisanie wartości po jej wcześniejszym sformatowaniu

Zadanie 1.
Utwórz w PHP program, który wypisze twoje imię i nazwisko, oraz adres.

```
<html>
<head>
</head>
<body>
<? //wypisujemy imię i nazwisko, używajac funkcji “print”
print ”Nazywam się Marian Kulikowski”;
echo ‘mieszkam na ulicy Bukowej’;
?>
</body>
</html>
```

Zadanie 2.
Popraw napisany skrypt tak aby Imię i nazwisko, oraz adres było pogrub., oraz napisane w osobnych wierszach.

4. Formatowanie powstającego kodu HTML

Naciśnięcie klawisza **ENTER** powoduje, że kod HTML również będzie zajmował dwa wiersze
Znak nowej linii (\n) !!! - (umieszczany tylko pomiędzy cudzysłowami (formatuje kod html – przejście do nowej linii)

Zadanie 3.
Popraw napisany skrypt tak aby wyświetlany tekst był napisany czcionką Verdana o wielkości 5 (1-7)
Dodatkowo imię i nazwisko oraz adres było pogrubione pochylone. Powstający kod html ma być przejrzysty. (w osobnych wierszach) użyj (\n)

5. Zmienne

zmienna – to coś, w czym możemy przechowywać pewną wartość

W skryptach PHP wykorzystanie zmiennych jest dużo prostsze niż w wielu innych językach. Nazwy zmiennych zawsze zaczynają się od znaku $. Nie jest konieczne deklarowanie zmiennych, tak jak ma to miejsce w wielu innych językach. Nazwy zmiennych mogą być dowolną kombinacją liter cyfr i znaków podkreślenia, pierwszy znak po znaku dolara nie może być cyfrą. Ważna wielkość liter. Do nadawania zmiennym określonych wartości służy znak (=) Aby stworzyć zmienną wystarczy nadać jej jakąś wartość, np:

```
$a=7;
$b="Jakiś tekst";
$c=2.654;
$d=0.0
```

Typy zmiennych. PHP obecnie obsługuje następujące typy zmiennych:

- integer - liczba całkowita
- double - liczba rzeczywista
- string - tekst
- array - tablica
- object - złożone zmienne definiowane przez użytkownika
- typ nieokreślony

Typ zmiennej jest określany automatycznie na podstawie przypisywanej wartości. I tak w powyższym przykładzie $a ma typ integer, $b ma typ string a $c i $d mają typ double (0 jest co prawda liczbą całkowitą, ale każda liczba zawierająca kropkę jest traktowana jako rzeczywista).
Jak widać na powyższym przykładzie, tekst powinien być zawsze ujęty w cudzysłowy. Jeżeli chcemy w tekście umieścić cudzysłów, należy poprzedzić go znakiem \. To samo dotyczy znaku $. W celu umieszczenia wewnątrz tekstu znaku \ należy napisać \\. Aby umieścić w tekście znak nowej linii można użyć sekwencji \n. Wewnątrz tekstu można też użyć zdefiniowanych wcześniej zmiennych:

```
$a=3;
$b="Jakaś wartość";
$c="$a, $b";
```

Zmienna $c będzie miała wartość "3, Jakaś wartość".

6. Łączenie ciągów

Łączenie ciągów to proces dodawania jednego ciągu do drugiego, operator złączenia (. kropka)

Zadanie 4.
Utwórz zmienne, w których zapamiętasz swoje imię i nazwisko. Następnie wypisz je. Użyj operatora złączenia (.), Zastosuj apostrof i cudzysłów.

```
<?
$imie = „Marian”
$nazwisko=”kulikowski”

print „ Mam na imie $imie” . ‘a nazywam się $nazwisko’;
?>
```

7. Apostrof kontra cudzysłów

Nazwy zmiennych zawartych w cudzysłowach zastępowane są przez ich wartości, natomiast w łańcuchach zawartych w pojedynczych cudzysłowach taka zamiana nie występuje.

Funkcja echo (ale nie print) pozwala przesyłać do przeglądarki kilka niezależnych ciągów danych.

Przykład:
echo ‘witaj ‘ , $imie

Łączenie zmiennych – cd.

```
$marka_auta=”Fiat”;
$model_auta=”Punto”;
$model_silnika=”1.1 SE”;
$spacja=” ”;
$moje_auto=$marka_auta . $spacja . $model_auta . $spacja . $model_silnika;
echo ‘moje auto to : ‘, $moje_auto;
```

Zadanie 5.
Umieść w osobnych zmiennych swoje dane: imie, nazwisko, ulica, miasto. Nastepnie utwórz 2 zmienne: przechowujące imie i nazwisko oraz zmienna adres przechowujaca ulice i miasto. Wyświetl dane za pomocą funkcji echo (,)

Proste operacje matematyczne:

Operator	*Działanie*
+	*Operator dodawania*
*	*Operator mnożenia*
-	*Operator odejmowania, także liczby ujemne*
/	*Operator dzielenia*
%	*Operator modulo – reszta z dzielenia całkowitego np. 8%5 = 3*
++	*Inkramentacja – zmienna jest zwiększana o 1*
--	*Dekramentacja- zmienna jest zmniejszana o 1*
+=	*Przypisuje do wyniku sumę bieżącej wartości zmiennej docelowej i argumentu*
-=	*Przypisuje do wyniku rożnicę bieżącej wartości zmiennej docelowej i argumentu*
.=	*Przypisuje do wyniku (będącego łańcuchem) jego połączenie z łańcuchem (będącym argum.)*

Zadanie 6.
Napisz skrypt, który wypisze Twoje informacje adresowe, używając operatora (.=) cw. 3.6
Wykorzystaj jedną zmienną przypisując jej kolejno poszczególne informacje.

Zadanie 7.
Utwórz skrypt, który wypisz pole i obwód prostokąta o zadanych bokach: np. a=12, b=20. Wypisz dane i wyniki w osobnych wierszach, wyniki wytłuść, sformatuj powstały kod html.

```
$boka=12;
$bokb=20;
print "<br>Bok a wynosi:". " " . $boka . "<br>\n";
print "Bok b wynosi:". " " . $bokb . "<br>\n" ;
print "Pole prostokąta wynosi: ". $boka*$bokb. "<br>\n" ;
print "Obwód prostokąta wynosi: ". 2*($boka+$bokb)."\n" ;
```

Dodawanie zmiennej do samej siebie

```
$suma=20;
$chleb=2;
$suma=$suma+$chleb;
```

Stara wartość suma jest powiększona o wartość $chleb

```
$suma=$suma + 1;
$suma++;
$suma+=2;  $suma=suma+2;
```

Funkcje round() i number_format()

```
$n=3.14;
$n=round($n); // w wyniku 3 (do liczby całkowitej)  lub
$n=3.142857;
$n=round($n, 3) // w wyniku da 3.143
```

Funkcja number_format() (wyświetla liczbę w postaci bardziej przyjaznej – wstawia przecinek jako separator tysięcy. Np.

```
$n=20943;
$n=number_format($n);  // 20,943
lub
$n=20943;
$n=number_format($n,2); //      20,943.00
```

8. Stałe

Stałe to specjalny typ danych. Przez cały czas przechowują wartości, które nadano im w chwili uruchomienia. Do tworzenia stałych wykorzystuje się słowo kluczowe define().

define('NAZWA', 'wartosc');

Nazwy stałych lepiej pisać wielkimi literami.

Zadanie 6
Napisz skrypt, który wyświetli tytuł, ilość, cenę, stawkę podatku (22 %) oraz wartość zakupionych książek w cenie brutto po uwzględnieniu stałego rabatu w wysokości 5 zł. Przy pisaniu skryptu użyj stałych VAT i RABAT, zastosuj funkcję number_format. Sformatuj powstały kod HTML.

Tytuł książki: : **Po prostu PHP**
Cena książki: **19.90 zł**
Ilość książek: **5**
Wartość podatku VAT: **0.22 %**

Wartość książek netto wynosi: **99.50** zł
Wartość podatku VAT wynosi: **21.89** zł

Razem do zapłaty:121.39 zł
Kupon rabatowy: **5 zł**

Po uwzględnieniu rabatu do zapłaty : 116.39 zł

KOD PHP:

```
$cena_ksiazki=19.90;
$cena_ksiazki=number_format($cena_ksiazki,2);
$ilosc_ksiazek=5;
define("VAT",0.22);
define("RABAT", 5);

print "<i>Tytuł książki: </i>: <b>Po prostu PHP</b><br>\n";
print "<i>Cena książki: </i>"." <b>".$cena_ksiazki." zł</b><br>\n";
print "<i>Ilość książek: </i>". " <b>".$ilosc_ksiazek.'</b><br>\n';
print "<i>Wartość podatku VAT:</i> "."<b> ". VAT ." %</b><br><br>\n";

print "<i>Wartosc ksiazek netto wynosi:</i>" . "<b> ". number_format($cena_ksiazki*$ilosc_ksiazek,2) ." </b>zł<br>\n";
print "<i>Wartość podatku VAT wynosi: </i>" . "<b> " . number_format($cena_ksiazki*$ilosc_ksiazek*VAT,2) ."</b> zł <br>\n";
print "<b>Razem do zapłaty:" . number_format($cena_ksiazki*$ilosc_ksiazek+$cena_ksiazki*$ilosc_ksiazek*VAT,2) ."</b> zł <br>\n";

print "<i>Kupon rabatowy: </i>" . "<b> " . RABAT . " zł </b><br><br><br>\n" ;
print "<h3>Po uwzględnieniu rabatu do zapłaty : ". number_format($cena_ksiazki*$ilosc_ksiazek+$cena_ksiazki*$ilosc_ksiazek*VAT-
RABAT,2). " zł </h3>\n";
```

9. Konwersje zmiennych:

```
$typ_silnika="3.0 SE";
$podatek=12;
$wartosc_podatku=$typ_silnika*$podatek;

echo „Wartosc podatku wynosi ", $wartosc_podatku; //          da wynik 36
```

Rzutowanie typów:

a) **nadanie typu zmiennej w chwili tworzenia**

```
$nowa_zmienna = 13;
$nowa_zmienna = (string) $nowa_zmienna; // zmiana wartości numerycznej na ciąg
$nowa_zmienna = (integer) $nowa_zmienna; // zmiana na liczbe
```

b) **sprawdzenie typu zmiennej - gettype($nazwa_zmiennej);**

```
$numer=5
echo gettype($numer);  // da w wyniku integer
```

b) **określenie typu zmiennej settype($nazwa_zmiennej, „string”);**
$numer=10;
settype($numer, „string”;

aby sprawdzić, że typ uległ zmianie możemy wyświetlic echo gettype($numer); // wyświetli string

c) **isset($numer)** - sprawdzenie czy podana zmienna została utworzona (czy istnieje)
echo isset($numer); // zwraca 1 gdy istnieje, gdy zmienna nie istnieje nie zwraca żadnej wartości

d) **unset($numer)** – usuwa zmienną oraz jej wartość

e) **echo empty($numer);** - odwrotność funkcji isset()
zwraca 1, gdy zmienna nie istnieje lub ma wartość 0, funkcja nie zwraca żadnej wartości gdy zmienna istnieje.

10. Zmienne środowiska – predefiniowane zmienne

$PHP_SELF – wyświetla informację o wykonywanym skrypcie (zwraca nazwę pliku)
$HTTP_USER_AGENT – określa używany system operacyjny i wersję przeglądarki (klienta)
$REMOTE_ADDR – zawiera adres IP użytkownika oglądającego naszą stronę

11. Pobieranie danych od użytkownika – obsługa formularzy

Formularz w HTML

<form action=nazwa_skryptu.php method=get>
…
</form>

atrybut method określa sposób w jaki dane zostaną wysłane do serwera.
Metoda GET nakazuje przegladarce aby dołączyła wartości jakie użytkownik umieścił w formularzu do adresu URL. Aby to zrealizować przeglądarka dodaje znak zapytania w celu oznaczenia końca właściwego adresu i początku danych. Dane są dołączane w postaci **nazwa-wartość** Więcej par oddzielone znakiem &, spacje są oddzielone znakiem +.
Wady to: dane są jawnie przesyłane, ale taka strona może być dodana do ulubionych, zapamiętanie stron przez wyszukiwarki sieciowe.
Metodą POST można przesłać większe ilości danych.

Zadanie 7.
Utwórz program, który wyświetli formularz z jednym polem tekstowym, a po wysłaniu tego formularza wyświetli wpisaną wartość. (Pole podaj imię, skrypt wyświetla Witaj ...imię)

Zadanie 8.
Uzupełnij skrypt z poprzedniego zadania o wielowierszowe pole (TEXTAREA – o szerokości 50 i 5 kolumnach) (Pole: Jakie są twoje ulubione witryny ?)

Zadanie 9.
Uzupełnij skrypt z poprzedniego zadania o pole wyboru (Checkbox), Pole: Czy jadłeś kiedyś kawior? (zobacz odpowiedz w zmiennej – domyślna opcja on)

Zadanie 10.
Popraw skrypt z poprzedniego zadania wstaw wiele pól wyboru (Checkbox), Twoje zainteresowania ? (pola sport, muzyka, ksiazka, internet). W yświetl odpowiedz.

Zadanie 11.
Dodaj do skryptu z poprzedniego zadania przyciski opcji (Radio). Płeć.
W odpowiedzi wyświetl. Jesteś ...dziewczyna ..chlopak

Zadanie 12.
Dodaj do skryptu z poprzedniego zadania pole listy z wyborem wieku. Przedziały wieku: poniżej 20 lat 20 – 30 lat, 30 – 40 lat, 41 i więcej.., Wyświetl odpowiedź.

12. Wyrażenia warunkowe i operatory

Wyrażenia warunkowe podobnie jak zmienne są podstawą każdego programu komputerowego.
W dynamicznych stronach www stosuje się je bardzo często, ponieważ pozwalają one dostosować przebieg działania programu (skryptu) do bieżących okoliczności. W PHP istnieją trzy najważniejsze słowa kluczowe umożliwiające budowanie wyrażeń warunkowych. `if, else, elseif`.

Każde wyrażenie takie zawiera klauzulę:

```
if (warunek) {
// wykonaj jakąś akcję
}
```

możemy ją rozbudować do postaci:

```
if (warunek) {
// wykonaj jakąś akcję
} else { // w przeciwnym wypadku
 // zrób coś innego !
}
```

a nawet:

```
if (warunek1) {
// wykonaj jakąś akcję
} elseif  (warunek2) {
// zrób coś innego!
} else {
 // zrób coś innego !
}
```

Jeżeli warunek jest spełniony, zostanie wykonany kod zawarty w nawiasach {} . Jeżeli nie PHP pójdzie dalej i sprawdzi kolejne warunki. Słowo kluczowe else można traktować jako domyślne działanie programu, które zostanie podjęte, jeżeli żaden z warunków nie zostanie spełniony.

W języku PHP warunek uznajemy za spełniony jeżeli:

- ma on postać zmiennej o wartości różnej od zera, od pustego łańcucha znaków i różnej od NULL
- ma on postać wyrażenia isset($var), a wartością zmiennej $var nie jest NULL, może to być natomiast 0 lub pusty ciąg znaków).

Istnieje też możliwość tworzenia bardziej złożonych wyrażeń logicznych za pomocą nawiasów, operatorów porównania i operatorów logicznych.

Operatory porównania i operatory logiczne:

Symbol	Znaczenie	Przykład
=	przypisuje się wartość równą	`$n = 1`
==	równa się	`$x == $y`
!=	nie równa się	`$x != $y`
<	mniejsze niż	`$x < $y`
>	większe niż	`$x > $y`
<=	mniejsze lub równe	`$x <= $y`
>=	większe równe	`$x >= $y`
!	negacja (typ logiczny)	`!$x`
&&	i (typ logiczny)	`$x  && $y`
||	lub (typ logiczny)	`$x || $y`

Zadanie 13.
Napisz prosty skrypt, w którym PHP losuje liczbę z zakresu od 1 do 10 a ty starasz się to liczbę odgadnąć.
Wskazówka**: Skorzystaj z funkcji **rand()

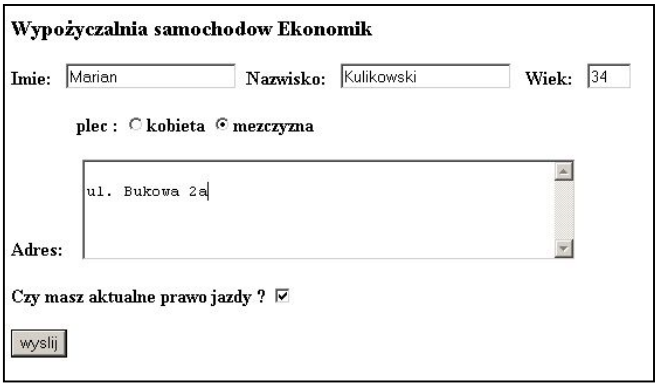
***Funkcja rand()** – w nawiasach podajesz początek i koniec przedziału oddzielone przecinkami, losowo jest generowane liczba łącznie z granicami przedziału np. rand(1,10)*

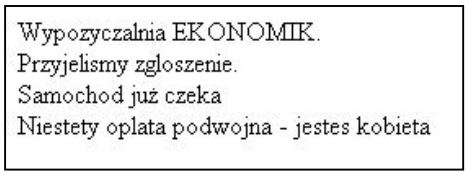
```
<html><body>  <?
$losowa=rand(1,10);
if ($liczba<$losowa){
echo "Liczba za mala";
echo "<br>Pomyslalem o $losowa, niestety nie ";
}
if ($liczba>$losowa){
echo "Liczba za duza";
echo "<br>Pomyslalem o $losowa, niestety nie ";
}
echo "wygrales !!";
?></body></html>
```

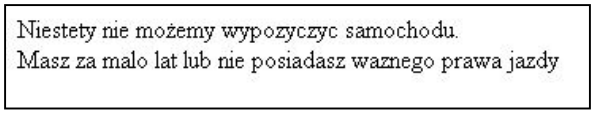
Zobacz strona 137 „Po prostu PHP”

wyjaśnienie logicznego OR AND !!!

Zadanie 14.
Napisz skrypt za pomocą którego firma wypożyczająca samochody może sprawdzić, czy klient może wypożyczyć samochód. Aby mógł to zrobić klient musi mieć aktualne prawo jazdy i co najmniej 21 lat.
Zadanie 15.
Popraw skrypt z ćwiczenia 14. Jeżeli przyjęliśmy zgłoszenie a klientem jest kobieta, w odpowiedzi dopisz wiersz: Oplata podwójna – jesteś kobietą.







13. Weryfikacja danych pochodzących z formularzy

Do weryfikowania danych wykorzystuje się wyrażenia warunkowe, funkcje, operatory i szereg innych konstrukcji językowych. Jedną z najczęściej stosowanych jest funkcja isset(), która sprawdza, czy danej zmiennej przypisano jakąkolwiek wartość. Może to być nawet wartość 0, ale nie NULL, czy False.

```
if   ( isset ($var ) )   {
// $var ma jakąś wartość
} else {
//   $var nie ma żadnej wartości
}
```

Sprawdza czy zmienna istnieje

isset() – zwraca wartość TRUE nawet wówczas, gdy wartością zmiennej jest pusty ciąg znaków.

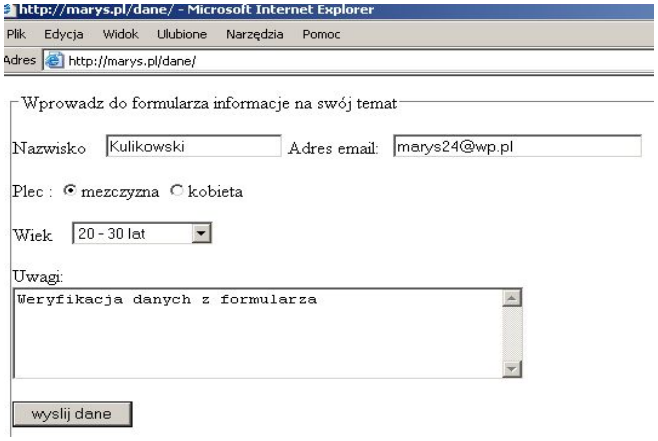
Funkcja **strlen()** – zwraca liczbę znaków w łańcuchu

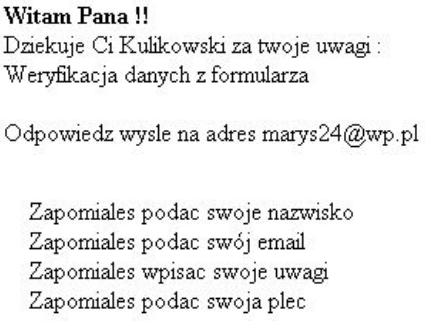
```
if   ( strlen($var) > 0 )   {
// $var ma jakąś wartość
} else {
//   $var nie ma żadnej wartości
}
```

Sprawdza czy zmienna ma jakąkolwiek wartość

Najlepszym sposobem na sprawdzenie, czy dane pole tekstowe zostało wypełnione, jest zbadanie długości łańcucha znaków.
Funkcja stripslashes (nazwa zmiennej) – usuwa znaki odwrotnego ukośnika

Zadanie 16. Weryfikacja danych





Exit - zatrzymuje gwałtownie wykonywanie programu (nie domyka żadnych znaczników).

HTMLSpecialChars("To nie zostanie pogrubione");
HTMLSpecialChars($nazwisko); - funkcja konwertuje znaczniki HTML na tekst do wyświetlenia, chroni przed wykonywaniem skryptów.

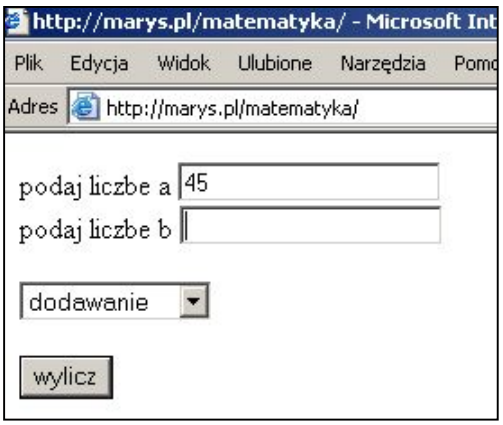
Sprawdzenie danych pochodzących z formularza.

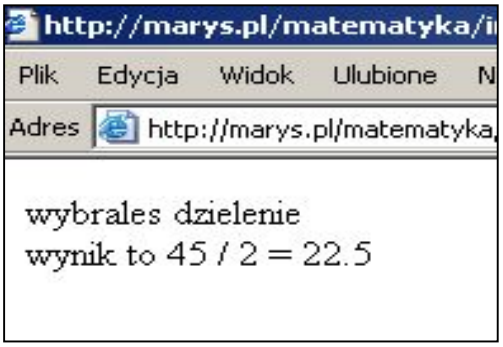
```
<html>
<head>
</head>
<body>

<?
if (strlen($nazwisko)>0) {
}
else {
$nazwisko=NULL;
print "Zapomiales podac swoje nazwisko <br>";
}
if (strlen($email)>0) {
}
else {
$email=NULL;
print "Zapomiales podac swój email <br>";
}
if (strlen($uwagi)>0) {
}
else {
$uwagi=NULL;
print "Zapomiales wpisac swoje uwagi <br>";
}

if (isset($plec)) {
  if ($plec==kobieta) {
  $wiadomosc="Witam Pania !!";
   }
   elseif($plec==mezczyzna) {
   $wiadomosc="Witam Pana !!";
   }
   }
else {
$plec=NULL;
print "Zapomiales podac swoja plec" ;
}

if ($nazwisko && $email && $uwagi && $plec) {
print "<b>".$wiadomosc."</b><br>"  ;
print "Dziekuje Ci ".$nazwisko. " za twoje uwagi : <br>". $uwagi;
print "<br> <br> Odpowiedz wysle na adres ". $email;
}
?>
</body></html>
```

```
<html>
<body>

<?

if ($nazwisko=="") {
print "Zapomiales podac swoje nazwisko";
exit;
}
if ($email=="") {
print "Zapomiales podac swój email";
exit;
}
if ($uwagi=="") {
print "Zapomiales wpisac swoje uwagi";
exit;
}

if ($plec!=kobieta && $plec!=mezczyzna) {
print "Zapomiales podac swoja plec";
exit;
}
else {
print "Witam ". $nazwisko;
}
?>
</body></html>
```

Zadanie
Napisz program wykonujący działania matematyczne

```
<html>
<?
if ($wyslano!=tak) {

print "
<form method=\"post\" action=\"". $PHP_SELF  ."\">";
?>

podaj liczbe a <input type=text name=a><br>
podaj liczbe b <input type=text name=b><br>
<input type=hidden name=wyslano value=tak>
<br>
<select name=znak>
        <option value=dodawanie>dodawanie
        <option value=odejmowanie>odejmowanie
        <option value=mnozenie>mnozenie
        <option value=dzielenie>dzielenie
</select>
<br><br>
<input type=submit value=wylicz></form>

<?
      }
  else {

      if($a=="" || $b=="") {

            if($a=="") {
                  print "nie podales a";
            }
            else {
                  print "nie podales b";
            }
            }
      else {
            if ($znak==mnozenie)
            {
            print "mnozenie ".$a*$b;
            }
            elseif ($znak==odejmowanie)
            {
            print "odejmowanie ";
            print $a-$b;
            }
            elseif ($znak=="dodawanie")
            {
            print "dodawanie ";
            print $a+$b;
                  }
            elseif ($znak==dzielenie && $b!=0)
            {
            print "dzielenie ". $a/$b;
            }
            else {
            print "nie dziel przez zero!!!!" ;
            }

      }

  }
?>
```

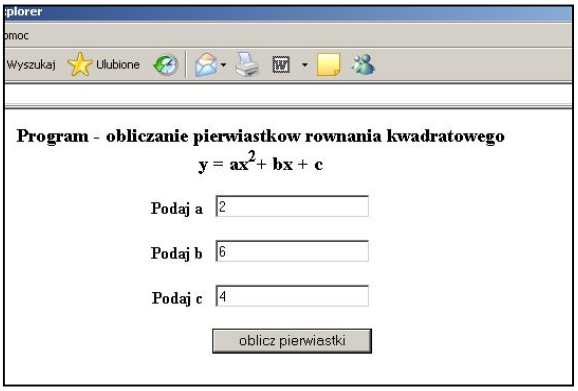


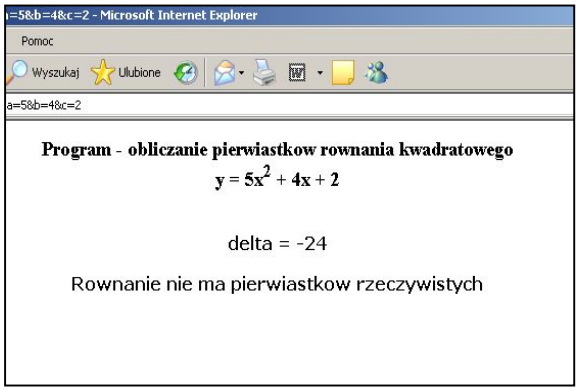


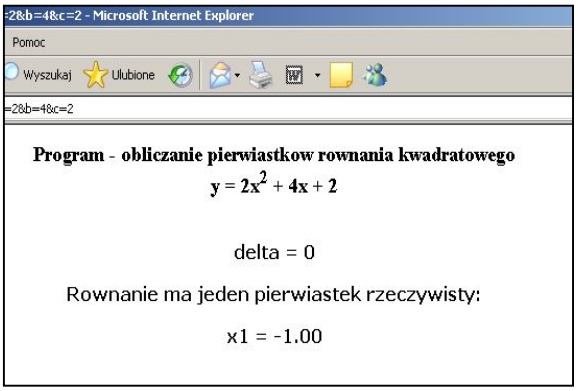
Zadanie 17

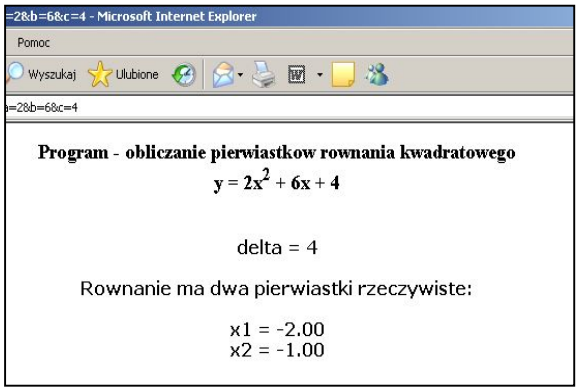
Napisz program, który wylicza pierwiastki równania kwadratowego o współczynnikach

$$y = ax^2+bx+c$$









Wskazówka: Pierwiastki równania kwadratowego oblicza się w zależności od delty:

$$\Delta = b^2 - 4 * a * c$$

$\Delta < 0$ => równanie nie ma pierwiastków rzeczywistych

$\Delta = 0$ => jeden pierwiastek $x1 = \frac{-b}{2 * a}$

$\Delta > 0$ => dwa pierwiastki $x1 = \frac{-b - \sqrt{\Delta}}{2 * a}$ $x2 = \frac{-b + \sqrt{\Delta}}{2 * a}$

Do sformatowanie wyników użyj funkcji number_format().

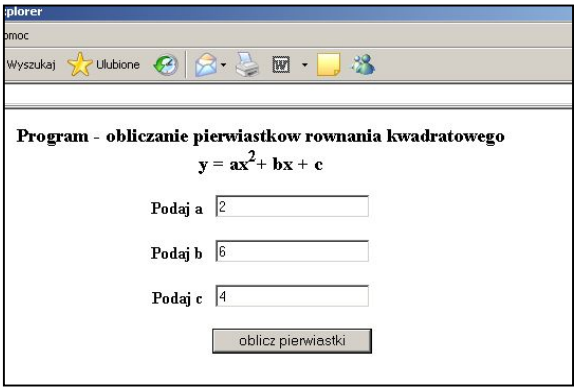
Popraw napisany program tak aby w przypadku:

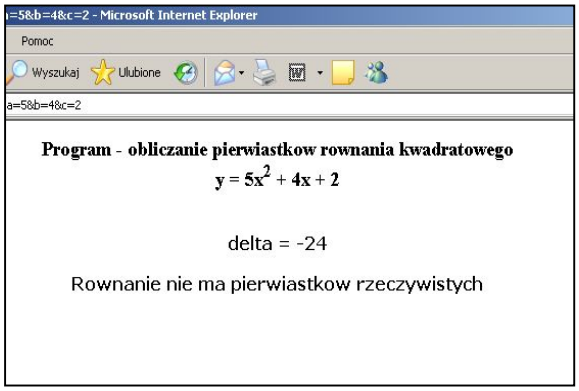
- **nie podania wartości współczynnika a obliczał równanie liniowe y = bx+c**
  jedno rozw. $x1 = \frac{-c}{b}$
- **nie podania wartości współczynnika a i b wypisał równanie y = c (brak pierwiastków)**
- **nie podania wartości współczynnika a, b, c poinformował o nie wpisaniu danych i wyświetlał formularz bądź odnośnik do strony z danymi.**

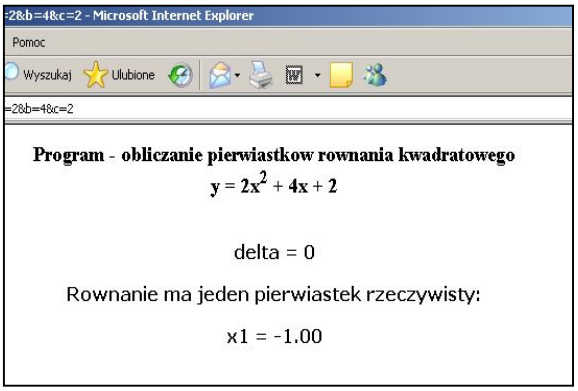
Zadanie 17

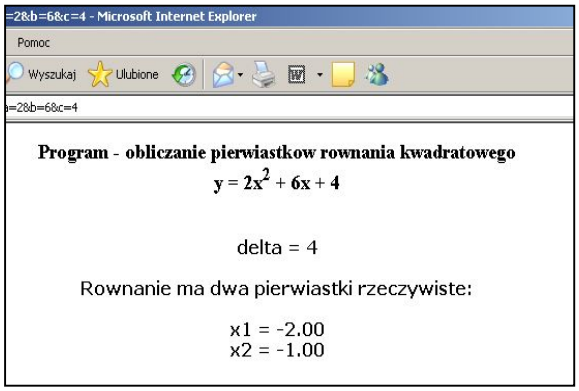
Napisz program, który wylicza **pierwiastki równania kwadratowego** o współczynnikach

$$y = ax^2+bx+c$$









Wskazówka: Pierwiastki równania kwadratowego oblicza się w zależności od delty:

$$\Delta = b^2 - 4 * a * c$$

$\Delta < 0$ => równanie nie ma pierwiastków rzeczywistych

$\Delta = 0$ => jeden pierwiastek $x1 = \frac{-b}{2 * a}$

$\Delta > 0$ => dwa pierwiastki $x1 = \frac{-b - \sqrt{\Delta}}{2 * a}$ $x2 = \frac{-b + \sqrt{\Delta}}{2 * a}$

Do sformatowanie wyników użyj funkcji number_format().

Popraw napisany program tak aby w przypadku:

- **nie podania wartości współczynnika a obliczał równanie liniowe y = bx+c**
  jedno rozw. $x1 = \frac{-c}{b}$
- **nie podania wartości współczynnika a i b wypisał równania y = c**
  (brak pierwiastków)
- **nie podania wartości współczynnika a, b, c poinformował o nie wpisaniu danych i wyświetlał formularz bądź odnośnik do strony z danymi.**

Wskazówka: Zagłębianie wyrażeń if ..else

Pierwszym naszym wyrażeniem if ..else sprawdzamy czy użytkownik wpisał jakikolwiek współczynnik a,b,c . W tym celu korzystamy z operatorów logicznych (| | - or).
Jeśli nie wypełnił żadnego pola a,b,c wypisujemy przy użyciu PHP pola do wpisania danych (formularz) i informujemy o nie wpisaniu danych.

Jeżeli nasz warunek zostanie spełniony czyli użytkownik poda przynajmniej jeden ze współczynników a, b ,c to :

- jeżeli wypełnił współczynnik **a** (podaj a) nasze równanie jest równaniem kwadratowym $y = a x^2 + bx + c$
  do obliczenia tego równania liczymy delte itd. (rozwiązania w zależności od delty)
  patrz **wskazówka obliczanie równania kwadratowego**

- jeżeli nie wypełnił współczynnika a **(podaj a)** a wypełnij współczynnik **b (podaj b)**
  nasze równanie jest równaniem liniowym y=bx+c

  Równanie liniowe posiada jedno rozwiązanie $x1 = \frac{-c}{b}$

- jeżeli nie wypełnił współczynnika a (podaj a) oraz nie wypełnij współczynnika b (podaj b) a wypełnij tylko współczynnik c (podaj c)
  nasze równanie ma postać y = c
  Równanie nie ma pierwiastków rzeczywistych.

Schemat programu

```
if ($a || $b || $c) {              //użytkownik wypełnił przynajmniej jedną pole
      if ($a) {                    //użytkownik podał a
            nasze równanie ma postać y=ax²+bx+c
            liczymy deltę;
            if (delta<0) {
                  brak rozwiązań
                  }
            elseif (delta==0) {
                  jedno rozwiązanie
                  }
            else {
                  dwa rozwiązania
                  }
      }
      elseif ($b) {                //użytkownik nie podał a,  podał b ...
            nasze równanie ma postać y=bx+c (równanie liniowe)
            rownanie ma jedno rozwiązanie
      }
      else {                       //użytkownik nie podał a i b, podał tylko c
            równanie ma postać y=c
            równanie nie ma rozwiązania
else {
nie wypełniłeś żadnego pola a,b,c
wyświetlamy formularz ponownie, lub przycisk wstecz
}
```

Rozwiązanie zadania 17

index.html

```
<html>
<head>
</head>
<body>
 <center>
 <h3>Program - obliczanie pierwiastkow rownania kwadratowego <br> y
= ax<sup>2</sup>+ bx + c</h3>
 <form metho=get action=oblicz.php>
 <b>Podaj a </b>   <input type=text name=a> <br><br>
 <b>Podaj b </b>   <input type=text name=b> <br><br>
 <b>Podaj c </b>  <input type=text name=c>  <br><br>
          
    
<input type=submit value="oblicz pierwiastki">
</form>
</center>
</body>
</html>
```

oblicz.php - bez sprawdzania wpisania a,b,c

```
<html>
<body>
 <center>
<h3>Program - obliczanie pierwiastkow rownania kwadratowego <br>

<?
print "y = $a"."x<sup>2</sup>"." + $b"."x + $c";
print "</h3> <br>";

$delta=$b*$b-4*$a*$c;
print "<font face=verdana> <b>";
print "delta = $delta <br><br>";

if ($delta<0)    {
   print "Rownanie nie ma pierwiastkow rzeczywistych";
   }
elseif ($delta==0) {
   $x1=-$b/(2*$a);

print "Rownanie ma jeden pierwiastek rzeczywisty: <br><br>" ;
    print "x1 = ". number_format($x1,2);
}

else {
    $x1=(-$b-sqrt($delta)) / (2*$a);
    $x2=(-$b+sqrt($delta)) / (2*$a);
    print "Rownanie ma dwa pierwiastki rzeczywiste: <br><br>";
    print "x1 = ". number_format($x1,2)." <br>";
    print "x2 = ". number_format($x2,2). " <br>";
}
?>
</b></font>
</center>
</body>
</html>
```

oblicz.php – wersja rozszerzona

```
<html>
<body>
 <center>
<h3>Program - obliczanie pierwiastkow rownania kwadratowego <br>

<?

if ($a || $b || $c) {

   if ($a) {

     print "y = $a"."x<sup>2</sup>"." + $b"."x + $c";
     print "</h3> <br>";

     $delta=$b*$b-4*$a*$c;
     print "<font face=verdana> <b>";
     print "delta = $delta <br><br>";

          if ($delta<0)    {
          print "Rownanie nie ma pierwiastkow rzeczywistych";
          }
          elseif ($delta==0) {
          $x1=-$b/(2*$a);

          print "Rownanie ma jeden pierwiastek rzeczywisty: <br><br>" ;
          print "x1 = ". number_format($x1,2);
          }

          else {
          $x1=(-$b-sqrt($delta)) / (2*$a);
          $x2=(-$b+sqrt($delta)) / (2*$a);
          print "Rownanie ma dwa pierwiastki rzeczywiste: <br><br>";
          print "x1 = ". number_format($x1,2)." <br>";
          print "x2 = ". number_format($x2,2). " <br>";
          }
    }
    elseif($b) {
     print "y = $b"."x + $c";
     print "</h3> <br>";
     print "rownanie liniowe <br><br>";
     $x1=-$c/$b;
     print "Rownanie ma jeden pierwiastek rzeczywisty: <br><br>";
     print "x1 = ". number_format($x1,2);
    }
    else {
     print "y = $c";
     print "<br><br> Rownanie nie ma pierwiastkow rzeczywistych";
      print "</h3> <br>";
    }
    }
else {

print "nie wpisales danych" ;

}

?>
</b></font>
</center>
</body>
</html>
```

14. Instrukcja wyboru [switch]

Aby sprawdzić wartość zmiennej i w zależności od wyniku wykonać różne działania, można wykorzystać złożoną instrukcję warunkową. :

```
if ($zmienna == wartosc1) {
      działanie1;
} elseif ($zmienna == wartość2) {
      działanie2;
} elseif ($zmienna ==wartość3) {
      działanie3;
} else {
      działanie4;
}
```

Zamiast tego można wykorzystać *instrukcję wyboru,* która pozwala na bardziej czytelne zaprogramowanie takiego działania. Oto postać tej instrukcji.:

```
switch (wyrażenie) {
 case wartość1:
      działanie1;
      break;
case wartość2:
      działanie2;
      break;
....
default:
      działanie;
      }
```

Jeżeli wyrażenie ma wartość1, wykonywane są instrukcje1, jeżeli wyrażenie ma wartość2 wykonywane są instrukcje2. Jeżeli nie uda się dopasować wartości wyrażenia wykonywane są instrukcje po słowie default.

Zadanie.
Napisz program, który wypisze bieżącą datę z miesiącem w języku polskim

Aby uzyskać date skorzystamy z funkcji `date`. Polską nazwę miesiąca okreslimy w funkcji switch.

Przykładowe znaczniki formatujące dla funkcji date.
d *– dzień miesiąca w formacie dwucyfrowym , z zerem na początku np. 02, 30*
m *– miesiąc w postaci dwucyfrowej z zerem na początku np. 05*
Y *– rok w postaci czteroliterowej np. 2004*

<html><body> Dzisiaj jest : <?

```
$dzien=date("d");
$miesiac=date("m");
$rok=date("Y");
```

```
switch ($miesiac) {
case '01': $miesiac="stycznia";
           break;
case '02': $miesiac="lutego";
           break;
case '03': $miesiac="marca";
           break;
......
default: $miesiac="niezidentyfikowany";
           break;
}
print "$dzien $miesiac $rok";
?>
</body></html>
```

15. Pętle i tablice

Mechanizm wykonywania powtarzających się czynności nazywamy pętlą.
W PHP występuje kilka rodzajów pętli. Są to pętle `while, do ...while, for, foreach.`

Pętle typu while:

```
while (warunek) {
instrukcje;
}
```

co rozumiemy jako: wykonuj instrukcję dopóki warunek jest spełniony.

Pętla do .. while jest odmianą pętli while o postaci:

```
do {
instrukcje;
}
while (warunek);
```

znaczenie w obu przypadkach jest takie samo. Jedyną różnicą jest fakt, że w przypadku pętli do ... while instrukcje zostaną wykonane co najmniej raz, nawet wtedy, gdy warunek jest od razu fałszywy.

Zadanie.
Przy użyciu pętli wypisz 10 razy zdanie: Nazywam się James, James Bond.

```
<html>
 <body>
 <?
$i=0;
while ($i<10) {
print "Nazywam się James, James Bond <br>";
$i++;
}
?>
</body></html>
```

Pętle for ..

Pętla `for` jest najczęściej używana, w przypadku gdy musimy wykonać fragment kodu określoną ilość razy. Daje ona możliwość określenia ilości iteracji pętli. Jej część warunkowa jest bardziej złożona od pętli `while` i składa się z trzech części.

```
for (ustawienie licznika pętli; sprawdzenie licznika pętli; zwiększenie licznika pętli) {
Wykonaj blok instrukcji
}
```

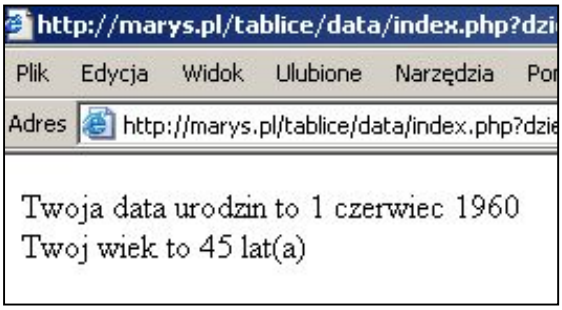
te trzy części pozawalają na tworzenie dosyć skomplikowanych warunków pętli.
Żadna z części nie jest obowiązkowa.

Zadanie.
Przy użyciu pętli for wypisz 10 razy zdanie: Nazywam się James, James Bond.

```
<html><body>
<?
for ($i=0; $i<10; $i++) {
 print "Nazywam się James, James Bond <br>";
}
?>
</body></html>
```

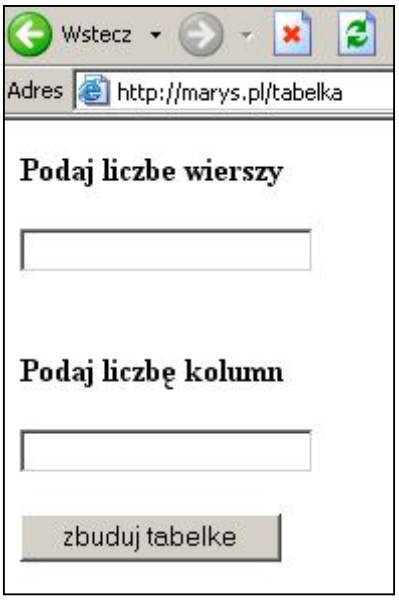
POKAŻ KONSTRUKCJE PRZYCISKÓW UKRYTYCH !!! (HIDDEN) np. niech wyświetlą na następnej stronie ilość wykonań pętli !!!!

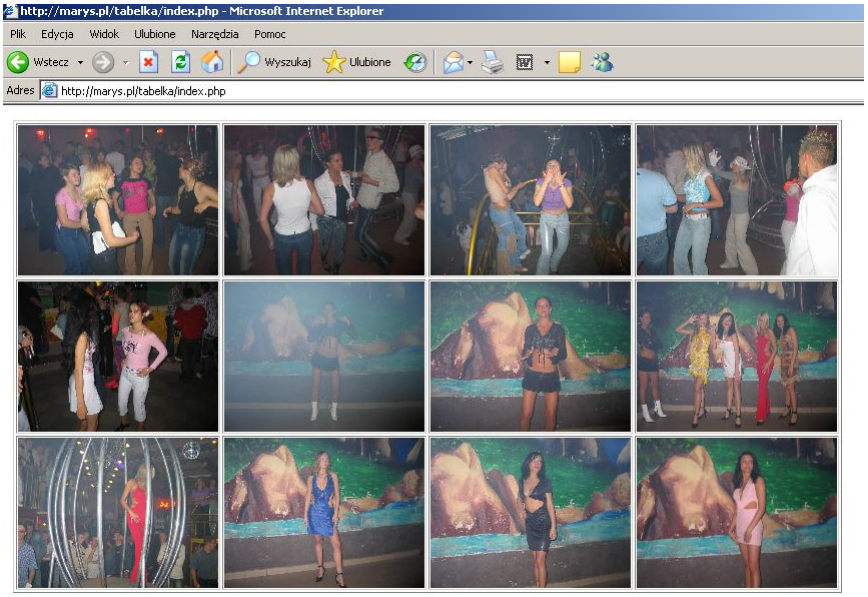
Zadanie




```
<html><body>
<?
if ($wyslano!="tak") {

       print "<h3>Wybierz date swoich urodzin</h3> ";
       print "<form method=get action=\"index.php\">";

       $miesiace=array("styczen", "luty",
"marzec","kwiecien","maj","czerwiec","lipiec","sierpien","wrzesien","pazdziernik","
listopad","grudzien");

       print "<select name=dzien>";
       $dzien=1;
       while($dzien<32) {

              print "<option value=$dzien>$dzien";
              $dzien++;
       }
       print "</select>";

       print  "<select name=\"miesiac\">";
       for ($i=0; $i<12;$i++) {
              print "<option value=$miesiace[$i]>$miesiace[$i]";
       }
       print "</select>";

       $rok=1956;
       print "<select name=rok>";
       do {
              print "<option value=$rok>$rok";
              $rok++;
       }
       while ($rok<2003) ;
       print "</select>";
       print "<br><br>";

       print "<input type=hidden name=wyslano value=tak>";
       print "<input type=submit value=\"wyslij dane\">";
       }
else {

       print "Twoja data urodzin to $dzien $miesiac $rok";
       $lata=2005-$rok;
       print "<br> Twoj wiek to $lata lat(a)";
}
?>
</body></html>
```

Zadanie. Tworzenie tabel






```
<html>
<body>
<?
if ($wstecz != "nie") {
print "<form action=\"".$PHP_SELF."\" method=\"post\">";
?>
<b>Podaj liczbe wierszy<br><br>
<input type="text" name="liczbaW">
<br><br><br>
Podaj liczbę kolumn <br><br>
<input type="text" name="liczbaK"><br><br>
                    <input type="hidden" name="wstecz" value="nie">
                    <input type="submit" value="zbuduj tabelke">
</b>
</form>
<?
}
else {
?>
<table border=1>
       <?
       $m=0;
       for ($i =0; $i < $liczbaW; $i++) {
                    print ”<tr>”;
                                 for ($j=0; $j<$liczbaK; $j++) {
                                        print " <td><img src=foto/img".$m.".jpg
                                        width=200> </td>";
                                 $m++;
                                 }
                    print ”</tr>”;
             }
?>
</table>
<?

}
?>
</body></html>
```

TABLICE

Tablice są zbiorem zmiennych posiadających tę samą nazwę, ale każda z nich ma inny indeks.
Każdy wpis w tablicy nazywany jest elementem.
Tablice można tworzyć identycznie jak zmienne, umieszczając tylko indeks w nawiasach kwadratowych.

Przypisywanie wartości początkowych do zmiennych tablicy jest nazywane inicjalizacją.

Inicjalizacja tablic:

- **bez wpisywania indeksów (PHP zrobi to za nas)**

```
$autor[]=”Adam Mickiewicz”;
$autor[]=”Juliusz Slowacki”;
```

- **przy użyciu jawnie podanych indeksów**

```
$stanyUSA[1]=”Washington”;
$stanyUSA[2]=”California”;
```

nie trzeba wpisywać ich po kolei, można opuścić tyle indeksów ile potrzebujesz

```
$stanyUSA[49]=”Alaska”;
$stanyUSA[13]=”Alabama”;
$stanyUSA[]=”Jaki indeks będzie ?”;
```

- **można zrezygnować z indeksów numerycznych** i używać ciągów znaków.
  Tego typu tablice są nazywane często **tablicami asocjacyjnymi**

```
$stolicestanyUSA[”ca”]=”Sacramento”;
$stolicestanyUSA[“il”]=Springfield”;
```

- **korzystanie z konstrukcji array()**

```
$autor=array(”Adam Mickiewicz” , ”Juliusz Slowacki”)
```

pierwszy pod indeksem zero ...
Jeżeli wypiszesz na ekran zawartość $autor[1], otrzymasz napis Juliusz Slowacki.
(nie ma ograniczenia wielkości tego typu tablic !!)

o**perator =>** pozwala na podanie początkowego indeksu tablicy

```
$StatesOfTheUSA=array(1=>”Alabama”, ”Alaska”, ”Arizona”, ”Arkansas”)
```

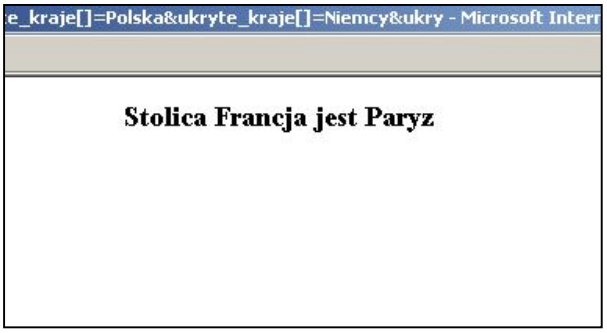
- **konstrukcja array(), operator (=>) i tablice asocjacyjne**

```
$stoliceUSA=array(“al”=>”Alabama”, “ak=>”Alaska”, “az”=>”Arizona”)
```

Przeglądanie tablic

Zadanie
Utwórz tablicę o nazwie uczniowie a następnie wyświetl jej zawartość przy użyciu pętli for oraz while.

Zadanie
Utwórz dwie tablice jedną przechowującą nazwy krajów (10 krajów) a drugą przechowującą stolice krajów.
Utwórz listę przewijaną z nazwami krajów, po wybraniu kraju skrypt ma pokazać odpowiednią stolicę kraju.



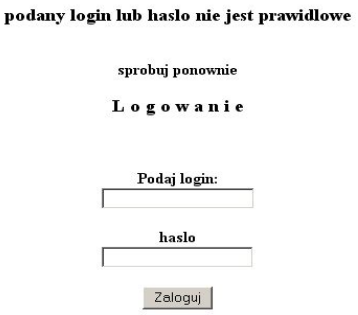
Plik index.php

```
<html> <body> <center> <h3>Jakiego kraju stolice chcesz znac ?</h3>

<form method=get action=stolica.php>
-------------------------------------------------------------------------------
<?
$kraje=array("Polska", "Niemcy", "Rosja", "Francja", "Anglia", "Holandia",
"Grecja", "United State of America");

print "<select name=kraj>";

      for ($i=0; $i<8; $i++) {
          print "<option value=$kraje[$i]> $kraje[$i] </option>";
      }
print "</select>";
                                  // druga pętla bo zamknięcie </select>
      for ($i=0; $i<8; $i++) {
          print "<input type=hidden name=ukryte_kraje[] value=$kraje[$i]>";
      }
?>
-------------------------------------------------------------------------------
<br><br>
<input type=submit value="pokaz stolice tego kraju">   </form>
</center></body></html>
```

Plik stolica.php

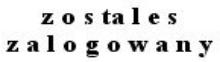
```
<html><body><center><h3>
-------------------------------------------------------------------------------
<?

$stolice=array("Warszawa", "Berlin", "Moskwa", "Paryz", "Londyn",
"Amsterdam","Ateny", "Washington");

for ($i=0; $i<8; $i++) {
    if ($kraj==$ukryte_kraje[$i]) {
        print "Stolica $kraj jest $stolice[$i]";
    }

}
 ?>
-------------------------------------------------------------------------------
 </h3></center></body></html>
```

Zadanie 1.



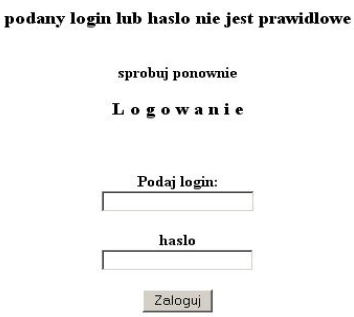
Opcja 1.

```
<html><body>
<?
if ($loguj!="tak") {
?>
      <center>
      <form method=get action=index.php>
      <h3>L o g o w a n i e</h3>    <br><br>
      Podaj login: <br>
      <input type="text" name="login" value="" />   <br>   <br>
      haslo  <br>
      <input type="password" name="haslo" value="" />   <br>    <br>
      <input type=hidden name="loguj" value="tak">
      <input type="submit" value="Zaloguj" />
      </form></center>
<?
}
else {
      $loginy=array("marys","jacek","lukes","asia");
      $hasla=array("marian88","nowakowski","kulikowski","chojnowska");

      $i=0;
            while ($loginy[$i]!="") {
                  if ($login==$loginy[$i] & $haslo==$hasla[$i]) {
                        print "<center>";
                        print "<h3>z o s ta l e s  <br> z a l o g o w a n
y</h3></center>";
                        $zalogowano="tak";
                  }
                  $i++;
            }
      if ($zalogowano!="tak") {
            print "<center><h3>podany login lub haslo nie jest
prawidlowe</h3><br>";
            print "<h4>sprobuj ponownie<h4></center>";
?>
            <center>
            <form method=get action=index.php>
            <h3>L o g o w a n i e</h3>          <br><br>
            Podaj login: <br>
            <input type="text" name="login" value="" />   <br>   <br>
            haslo  <br>
            <input type="password" name="haslo" value="" />   <br>    <br>
            <input type=hidden name="loguj" value="tak">
            <input type="submit" value="Zaloguj" />
            </form></center>
<?
}
}
?>
</body></html>
```



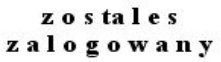
Opcja 2.

wykorzystanie tablic asocjacyjnych



index.html

```
<html>
<body>
<center>
<form name="Log" action="loguj.php" method="get">
<h3>L o g o w a n i e</h3>
<br><br>
Podaj login: <br>
<input type="text" name="login" value="" />   <br>   <br>

haslo  <br>
<input type="password" name="haslo" value="" />   <br>    <br>
<input type="submit" value="Zaloguj" />
</form>
 </center>
</body>
</html>
```



loguj.php

```
<html><body><center>
<?

$t_dane=array("marys22"=>"marian88", "marys24"=>"olek888", "lukes"=>"luk",
"marys25"=>"marian25");

if ($login && $haslo) {

      if($t_dane[$login]==$haslo)  {
            print "Jestes zalogowany twój login to <b>$login</b> a haslo :
<b>$haslo</b>";
            exit;
      }
      else {
            print "Niestety to nie to haslo lub login, sprobuj ponownie
<br><br>";
      }
}

else {
      print "Nie wprowadziles danych !!!";
     }
?>
<form name="Log" action="loguj.php" method="get">
<h3>L o g o w a n i e</h3><br><br>
Podaj login: <br><input type="text" name="login" value="" />   <br>   <br>
haslo  <br><input type="password" name="haslo" value="" />   <br>    <br>
<input type="submit" value="Zaloguj" />
</form></center> </body> </html>
```

Tablice – przeglądanie tablic – funkcje current() i key()

Funkcje current() key() next() prev()

Zadanie
Napisz tablicę z nazwiskami aktorów o indeksach 4, 1, 93, 24. i bez indeksu ..
Następnie odczytaj bieżący indeks tablicy za pomocą funkcji key(), następnie użyj funkcji next () i ponownie wyświetl bieżący indeks tablicy.

```
<?
$aktorzy[4]="Boguslaw Linda";
$aktorzy[1]="Marek Konrad";
$aktorzy[93]="Mel Gibson";
$aktorzy[24]="Tom Cruise";
$aktorzy[]="Marian Kulikowski";  //ten element zostanie zapisany w tablicy pod indeksem 94

$biezacy_indeks=key($aktorzy);
print "$biezacy_indeks";
print "<br>";
      next($aktorzy);

 $biezacy_indeks=key($aktorzy);
 $biezaca_wartosc_indeksu=current($aktorzy);
 print "$biezacy_indeks";
 print " - " ;
 print "$biezaca_wartosc_indeksu";
?>
```

Jeśli funkcją prev lub next przejdziesz za ostatni lub pierwszy element nie będziesz się mógł cofnąć.

Funkcje list() i each()

Zamiast tworzyć pętlę sprawdzającą ogromne obszary pustej przestrzeni, można skorzystać z funkcji `list() i each()`, które umożliwiają dostęp tylko do tych elementów tablicy które zawierają dane.

while (list(WartośćIndeksu , ZawartośćElementu)=each(NazwaTablicy))

```
<html><body>
<?
 $auta[23]="Fiat Croma";
 $auta[3]="Opel Vectra";
 $auta[14]="Mazda 323";
 $auta[65]="Skoda Fabia";
 $auta[5]="Maluch 126 p";
 $auta[]="Suzuki";
 $auta[6]="Porsche";

while (list($klucz, $wartosc_klucza)=each($auta))
{
    print "<br>";
    print $klucz. " - " . $wartosc_klucza;
}
?>
```

Sortowanie tablic

PHP posiada kilka funkcji pozwalających sortować tablice. Zapoznamy się z 5 najczęściej używanymi. Działają we współpracy z funkcjami list() i each(), opisanymi przed chwilą.

sort() – sortuje elementy tablicy w kolejności alfabetycznej.
Jako parametru wymaga nazwy tablicy do posortowania.

Przykład: Utwórz tablicę z nazwiskami.

```
$nazwiska[0]=”Kulikowski Marian”;
$nazwiska[1]=”Nowak Karol”;
$nazwiska[2]=”Antkowiak Janusz”;
$nazwsisk[3]=”Turnau Grzegorz”;
```

po użyciu funkcji sort($nazwiska);

```
$nazwiska[0]=”Antkowiak Janusz”;
$nazwiska[1]=”Kulikowski Marian”;
$nazwiska[2]=”Nowak Karol”;
$nazwsisk[3]=”Turnau Grzegorz”;
```

rsort() – sortuje elementy tablicy w kolejności alfabetycznej (w odwrotnym porządku).
Jako parametru wymaga nazwy tablicy do posortowania.

---

asort() – pobiera tablicę z indeksami w postaci ciągów i sortuje je według zawartości.

Przykład:

```
$nazwiska=array(„mk”=>”Marian Kulikowski”,”nk”=>”Nowak Karol”,
                  ”aj”=>”Antkowiak Janusz”,”tg”=>”Turnau Grzegorz”)

asort($nazwiska);

$nazwiska[”aj”]=”Antkowiak Janusz”;
$nazwiska[”mk”]=”Kulikowski Marian”;
$nazwiska[”nk”]=”Nowak Karol”;
$nazwsisk[”tg”]=”Turnau Grzegorz”;
```

Elementy są posortowane i mają swoje indeksy znakowe. (sort. według elementów) !

arsort() – to samo co asort() tylko w odwrotnym porządku

---

ksort() – zamiast zwracać tablicę asocjacyjną posortowaną według wartości, sortuje ją według indeksów

```
ksort($nazwiska);

$nazwiska[”aj”]=”Antkowiak Janusz”;
$nazwiska[”nk”]=”Nowak Karol”;
$nazwiska[”mk”]=”Kulikowski Marian”;
$nazwsisk[”tg”]=”Turnau Grzegorz”;
```

Funkcje implode() i explode()

implode("separator", $NazwaTablicy) – łączy wszystkie elementy w jeden ciąg.

Przykład:
```
$nazwa_ciagu=implode(„separator", $nazwa_tablicy);
```

explode("separator", $nazwa_ciagu);

- jeżeli mamy napis, który można podzielić na elementy w miejscach wystąpienia separatorów, na przykład myślnika, znaku & lub spacji, można skorzystać z tej funkcji.

```
Przykład:

$all_nazwiska="kulikowski-nowak-wisniewski-kowalski";
$nazwiska=explode("-", $all_nazwiska);

while(list($klucz, $wartosc)=each($nazwiska)) {
print "<br> $klucz $wartosc „;
}

wynik

0 kulikowski
1 nowak
2 wisniewski
3 kowalski
```

Pętle foreach – przeglądanie tablic

```
foreach ($NazwaTablicy As $ElementTablicy
{
wykonanie instrukcji w klamrach
}


lub

foreach ($NazwaTablicy As $IndeksTablicy => $ElementTablicy)
{
wykonanie instrukcji w klamrach
}
```

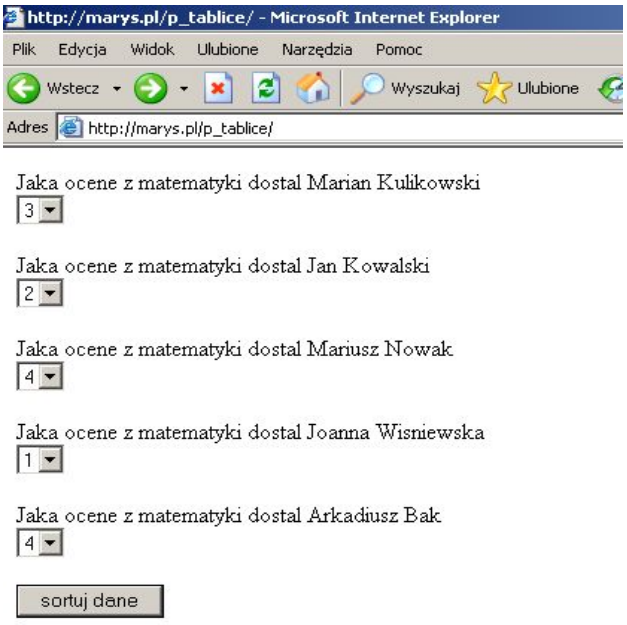
```
Przykład:
$nazwiska=array("kulikowski","nowakowski","wisniewski","kowalski");
foreach($nazwiska As $klucz => $wartosc) {
print  "<br> $klucz - $wartosc";
}
wynik:

0-kulikowski
1-nowakowski
2-wisniewski
```

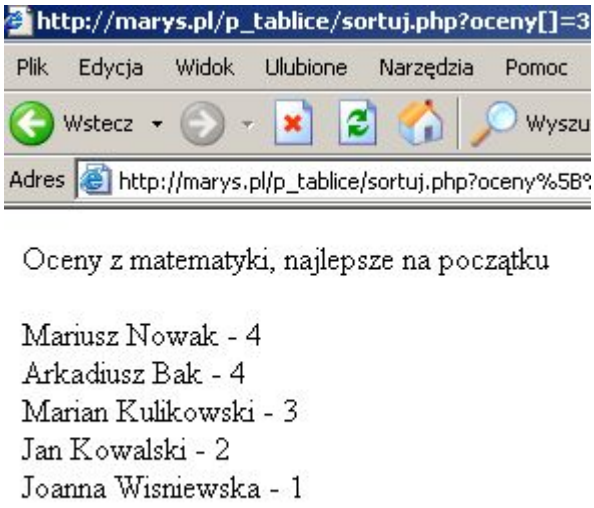
Praktyczne zastosowanie tablic – przykład użycia kilku funkcji tablic

Zadanie.
Wyświetl formularz z pytaniem uczniów o ocenę z egzaminu. (nazwiska uczniów zapamiętaj w tablicy).
Następnie wyświetl posortowane wyniki według najlepszych ocen.

```
<html><head></head><body><?
$studenci=array("Marian Kulikowski", "Jan Kowalski",
"Mariusz Nowak","Joanna Wisniewska", "Arkadiusz Bak");
?>
<form method=get action=sortuj.php>
<?
 while(list($klucz, $wartosc)=each($studenci)) {
 print "Jaka ocene z matematyki dostal $wartosc <br>";
 print "<select name=oceny[]>";
 ?>
 <option value=5> 5 </option>
 <option value=4> 4 </option>
 <option value=3> 3 </option>
 <option value=2> 2 </option>
 <option value=1> 1 </option>
 </select>
 <br><br>  <?
print "<input type=hidden name=studenci[]
value='$wartosc'>";
 }
?>
<input type=submit value="sortuj dane">
</form></body></html>
```

```
<?

print "Oceny z matematyki, najlepsze na początku
<br>";

while (list($klucz, $wartosc)=each($studenci))
 {
 $ocena_student[]=$oceny[$klucz].$studenci[$klucz];
 }

arsort($ocena_student);

while(list($klucz, $wartosc)=each($ocena_student))
{
print "<br>$studenci[$klucz] - $oceny[$klucz]";
}

?>
```

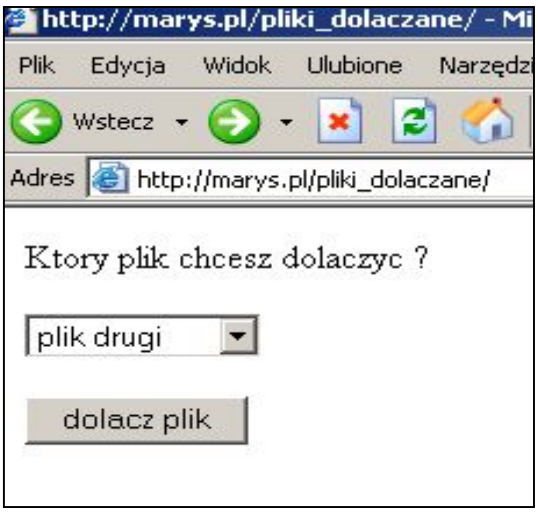




Pliki dołączane - include

```
include ("test.txt");

lub z użyciem zmiennych

include("test" . $name . ".txt");
```

Przykład:



Utwórz plik1.txt, plik2.txt

```
<html>
<head>
</head>

<body>
<form method=get action=dolacz.php>

Ktory plik chcesz dolaczyc ? <br><br>

<select name=numer>

<option value=1>plik pierwszy</option>
<option value=2>plik drugi</option>

</select>

<br><br>
<input type=submit value="dolacz plik">

</form>

</body></html>
```

```
<html>
<body>

troszke tekstu u góry <br><br>

<?

if ($numer <>"") {

        include("plik" .$numer. ".txt") ;
}

?>

<br>
troszke tekstu na dole <br>
</html>
```

Funkcje – definiowanie i wywoływanie funkcji

Funkcje w PHP definiujemy za pomocą słowa kluczowego function, po którym następuje nazwa funkcji oraz lista argumentów ujetych w nawiasy okrągłe. Schematycznie wygląda to następująco:
Parametry (argumenty), czyli dane przekazywane do funkcji, są widoczne wewnątrz niej pod odpowiednimi nazwami (takimi jakie zadeklarowaliśmy w nagłówku funkcji) i można na nich operować jak na innych zmiennych.

Jeżeli chcesz, by funkcja zwróciła jakąś wartość, należy posłużyć się instrukcją return. Powoduje ona zakończenie działania funkcji i zwrócenie jako wyniku wartości wyrażenia występującego w instrukcji return.

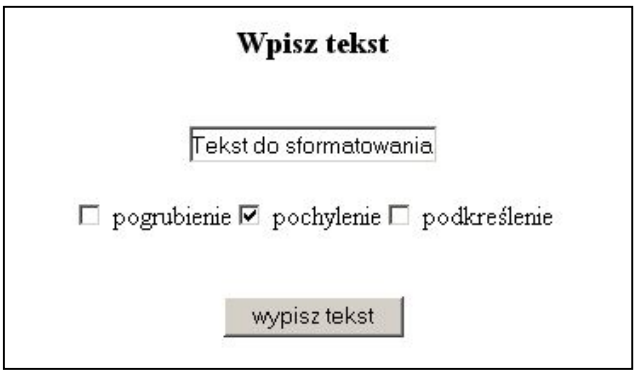
```
function nazwa_funkcji (arg1, arg2, ..., argn) {
instrukcje;
return wynik;
}
```

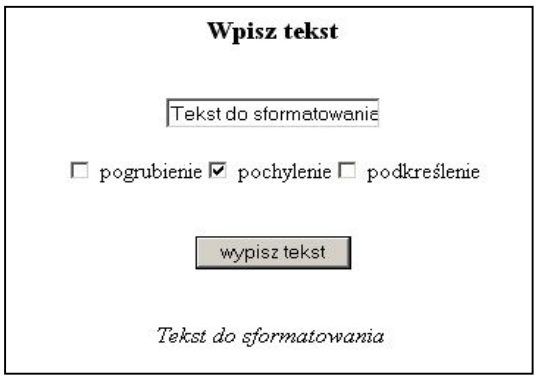
Argumenty funkcji – przekazywanie przez wartość

Zadanie
Napisz funkcję, która łańcuch sformatuje pogrubieniem.

```
<?
function pogrubienie($tekst) {
return "<b>" . $tekst . "</b>";
}
print "To jest tekst zwykly";
print  pogrubienie ("a to jest tekst pogrubiony.");
?>
```

Zadanie






```
<?
$sciezka="../";
include "../dolacz/gora.inc";
include "../dolacz/lewa.php";
print "<td valign=top width=470> ";
print "<br><br><br>";

function pogrubienie(&$lancuch) {
        if ($GLOBALS["gruby"]=="tak") {
                $lancuch= "<b>$lancuch</b>";
        }
        if ($GLOBALS["pochyly"]=="tak") {
                $lancuch="<i>$lancuch</i>";
        }
        if ( $GLOBALS["podkreslony"]=="tak") {
        $lancuch="<U>$lancuch</U>";
        }
return $lancuch;
}
```

```
print "<form method=get action=index.php>";
print "<center>";
print "<h3>Wpisz tekst </h3><br>";
print "<input type=text name=lancuch value=\"$lancuch\"> <br><br>";

print "<input type=checkbox name=gruby value=tak> pogrubienie";

print "<input type=checkbox name=pochyly value=tak> pochylenie";

print "<input type=checkbox name=podkreslony value=tak>
podkreślenie";

print "<br><br><br>";

print "<input type=submit value=\"wypisz tekst\">";
print "<br><br><br>";

print pogrubienie($lancuch);
include "../dolacz/koniec.inc";
?>
```

Przekazywanie przez zmienną (przez referencję)

Drugi sposób przekazywania argumentów do funkcji – przez zmienną. Aby zasygnalizować że chcesz użyć tej metody, musisz dodać znak & (ampersand) przed przekazywaną zmienną.
Zmienna zostaje zmieniona wewnątrz funkcji.
Przykład – powyżej

Ustawienia domyślnych wartości parametrów

```
function podatek ($podatek , $zarobki=2500) {
$zarobki=$zarobki-(($zarobki/100)*$podatek);
return $zarobki;
}
echo (podatek(25));
```

Zasięg zmiennej – zmienne lokalne i globalne

Zmienne utworzone poza funkcjami nadal są dostępne na całej stronie. Zmienne zapisane we wnętrzu funkcji są **zmiennymi lokalnymi**. Zmienne dostępne na całej stronie nazywamy **zmiennymi globalnymi.**

Użycie zmiennych globalnych we wnętrzu funkcji

```
1. po przez poprzedzenie nazwy zmiennej słowem global
      np. global $gruby;
2. poprzez skorzystanie z tablicy $GLOBALS
      np. $GLOBALS[”gruby”]
```

Utrzymywanie wartości przez zmienne lokalne

Przy każdym wywołaniu funkcji zmienna lokalna jest niszczona podczas jej zakończenia. Gdybyśmy np. chcieli coś zliczać, nie utrzymywanie wartości przez zmienną pomiędzy kolejnymi wywołaniami funkcji stanowiłoby duży problem.
Słowo kluczowe **`static`** powoduje, że linia jest wykonywana tylko podczas pierwszego wywołania funkcji i od tego czasu jest ignorowana. Np.

```
function  numer_wywolania() {
static $numer=0;
return $numer=$numer+1;
}
echo (numer_wywolania());  - wypisze 1
echo (numer_wywolania());  - wypisze 2
echo (numer_wywolania());  - wypisze 3
```

Funkcje daty i czasu

echo date(‘F j, Y’); //January 21, 2005

Zaczniki formatujące dla dat

Y	Rok wyrażony 4 cyframi	2005
F	miesiąc	February
j	Dzień miesiąca wyr 1 lub2 cyframi	21
l	Dzień tygodnia	Monday
g	Godzina w fromacie 12 godz	6
G	Godz w formacie 24 godzinnym	18
i	Minuty	45
a	Am lub pm	am

$dates=getdate();
echo $dates[‘month’]; //January

Tablica getdate()

yaer	rok	2005
mon	miesiąc	12
month	Nazwa misiąca	December
mday	Dzień miesiąca	25
weekday	Dzień tygodnia	Tuesday
hours	godziny	11
minutes	Minuty	56
seconds	sekundy	47

Zadanie
Program wyświetla w polu select aktualny dzień misiąc i rok, dodatkowo wyświtla date w formacie dzień tygodnia, godzina : sekunda
Pola select utworzyć za pomocą pętli, nazwy miesiąca zapisać w tablicy, tablice przeglądać za pomocą pętli **foreach.** Folder o nazwie kalendarz dołączyć do strony. Utworzyć funkcję kalendarz.


```
<?
$sciezka="../";

include "../dolacz/gora.inc";
include "../dolacz/lewa.php";
print "<td valign=top width=470> ";
print "<br><br><center>";

function kalendarz($this dzien,$this miesiac, $this rok) {

$miesiace=array(1=>"styczen","luty","marzec","kwiecien",
       "maj","czerwiec","lipiec","sierpien","wrzesien",
       "pazdziernik","listopad","grudzien");

print "<select name=\"dni\">\n";
       for ($i=1;$i<32;$i++) {
                      print "<option value=\"$i\"";
                             if ($i==$this_dzien) {
                                 print "selected=\"selected\"";
                             }
                      print "> $i </option>\n";
      }
print "</select>";

print "<select name=\"miesiac\">\n";
       foreach ($miesiace as $klucz => $wartosc) {
              print "<option value=\"$klucz\"";
                     if ($klucz==$this miesiac) {
                            print " selected=\"seleceted\"";
                     }
              print ">$wartosc</option>\n";
     }
print "</select>\n";


print "<select name=\"rok\">\n";
for ($i=1980;$i<2010; $i++) {
              print "<option value=\"$i\"";
                     if ($i==$this rok) {
                            print "selected=\"selected\"";
                     }
              print ">$i</option>\n";
   }
print "</select>";

}

$data=getdate();
echo $data['mday'];
echo " ";   echo $data['month'];  echo " ";
echo $data['year']; echo "<br><br>";

kalendarz($data['mday'], $data['mon'], $data['year']);

echo "<br><br>";
echo date('l, G:i a');
include "../dolacz/koniec.inc";
?>
```

Obsługa plików i katalogów

PHP posiada dwa zestawy funkcji operujących na plikach, pierwszy zestaw korzysta z uchwytu pliku, drugi korzysta bezpośrednio z nazwy pliku.

Uchwyt pliku jest prostą wartością całkowitą używaną do identyfikacji pliku na którym pracujesz aż do jego zamknięcia.

Otwieranie i zamykanie plików:

W czasie działania na plikach realizowane są zwykle trzy kroki:

- otwarcie pliku, na którym mamy zamiar pracować oraz przypisanie mu uchwytu pliku,
- czytanie lub zapis do pliku, korzystając z uchwytu pliku,
- zamknięcie pliku przy użyciu uchwytu

fopen() – służy do otwarcia pliku, za jej pomocą można otworzyć także połączenie z adresem URL
Zwraca wartość true gdy operacja powiodła się i false gdy błąd.

Jej deklaracja wygląda następująco:

```
fopen(filename, mode, use_include_path);
np.   $fp=fopen("./data.txt", "r");
```

filename – scieżka do pliku,

za pomocą parametru mode określamy, w jaki sposób będzie używany plik.
Możesz otworzyć plik do odczytu, do zapisu lub do dołączania.
Parametr **mode** może przybierać następujące wartości

Wartość	Opis
r	Otwarcie tylko do odczytu. Znacznik pozycji jest umieszczony na początku pliku
r+	Otwarcie do odczytu i zapisu. Znacznik pozycji jest umieszczony na początku pliku
w	Otwarcie tylko do zapisu. Istniejące dane są tracone, jeżeli plik nie istnieje PHP próbuje go utworzyć
w+	Otwarcie do zapisu i odczytu. Istniejące dane są tracone; jeżeli plik nie istnieje PHP próbuje go utworzyć
a	Otwarcie pliku do dołączania. Dane są dopisywane na końcu istniejącego pliku; jeżeli plik nie istnieje PHP próbuje go utworzyć
a+	Otwarcie do dołączania i odczytu. Dane są dopisywane na końcu istniejącego pliku; jeżeli plik nie istnieje PHP próbuje go utworzyć

fclose() – po zakończeniu pracy ze skryptem trzeba go zamknąć. Do tego służy fclose(). Jako parametr podajemy uchwyt do otwartego pliku.

np.
```
fclose($fp);
```

czytanie i zapis do pliku

fread() – funkcja używana jest do odczytywania ciągu znaków z pliku. Posiada dwa argumenty, **uchwyt pliku** i liczbę **length**. Odczytuje z pliku maksymalnie length bajtów i zwraca je w postaci ciągu.

```
np.    $fp=fopen("data.txt", "r");
       $data=fread($fp, 10);
```

spowoduje to odczytanie pierwszych 10 bajtów z pliku data.txt i przypisanie ich do zmiennej $data jako ciągu znaków.

Musimy zwrócić uwagę na następujące szczegóły.

- Załóżmy, że powtarzasz wywołania funkcji fread(). Pierwsze wywołanie przesuwa znacznik pozycji o 10 pozycji w kierunku końca pliku. Przy następnym wywołaniu nie odczytamy tych samych danych, ale kolejne 10 bajtów, rozpoczynając od ostatniej pozycji,
- Jeżeli pozostało do odczytania mniej niż 10 bajtów, fread() odczyta i zwróci wszystkie, które pozostały.

fwrite() – służy do zapisu danych do pliku. Wymaga ona dwóch argumentów, uchwytu pliku fp i ciągu znaków do zapisu. Spowoduje zapisanie zawartości drugiego argumentu do pliku wskazywanego przez uchwyt.

przykład:

```
$fp=fopen(”data.txt”, ”w”);
fwrite($fp, “Marian Kulikowski”);
```

jeżeli jako trzeci argument podamy liczbę całkowitą length, wstrzymamy zapis po length bajtach.

przykład – prosty licznik wizyt na stronie

```
<?

$plik_licz="./licz.txt";

$plik=fopen($plik_licz, "r");
if (!($plik)) {
print ("Nie mogę otworzyc pliku licz.txt");
}
else {
$liczydlo = (int) fread($plik,20);
$fclose($plik);

/* ponieważ fread() zwraca ciąg to musimy otrzymany wynik skonwertować na
liczbę, korzystamy z odpowiedniego rzutowania typów.
*/

$liczydlo++;
echo "jestes gosciem nr: $liczydlo";

$plik=fopen($plik_licz, "w");
fwrite($plik, $liczydlo);
fclose($plik);
}
?>
```