Funkcje, tablice, formularze PHP

rezultat projektu

Nowe Technologie wsparciem dla edukacji

nr umowy - POWERSE-2018-1-PL01-KA101-049291

realizowanego ze środków POWER na zasadach programu Erasmus+

sektor Edukacja szkolna

„Ponadnarodowa mobilność kadry edukacji szkolnej"

Zespół Szkół Łączności w Gliwicach

44-100 Gliwice, ul. Warszawska 35 tel. 32 231 36 12, **www.zsl.gliwice.pl**

przydatne strony

http://uazz.pl/index.php/php/115-php-funkcje

http://pl.wikibooks.org/wiki/PHP/Funkcje

http://pl.wikibooks.org/wiki/PHP

http://teleinfo.pb.edu.pl/krashan/files/tech_int_3_wyklad_3.pdf

http://home.agh.edu.pl/~horzyk/lectures/ahdydwww.php#php

Definicja funkcji w PHP

Informacje są przekazywane do funkcji przez listę parametrów, oddzielonych przecinkami.

```
function nazwaFunkcji($parametr1, $parametr2 )
  {
 // operacje na parametrach
  return $wartosc; // opcjonalnie, gdy funkcja ma zwrócić wartość
                   //instrukcja  return przerywa działanie funkcji.
  }
```

Aby skrypt był wykonywany dopiero po załadowaniu strony, można umieścić go w funkcji.

Funkcja zostanie wykonana przez wywołanie funkcji.

Można wywołać funkcję z dowolnego miejsca na stronie.

Nazwa funkcji musi się składać wyłącznie z liter, cyfr i znaków podkreślenia (ponadto nie może zaczynać się cyfrą). Nie można również tworzyć nazw funkcji kolidujących z nazwami wbudowanych funkcji php. Ponadto nie można zdefiniować dwóch funkcji o tej samej nazwie (wielkość liter nie ma znaczenia), nawet jeśli przyjmują różne zestawy parametrów.

Definicja funkcji w PHP

PHP obsługuje przekazywanie argumentów przez wartość (domyślnie), przez referencję i wartości domyślne argumentów.

Aby skrypt był wykonywany dopiero po załadowaniu strony, można umieścić go w funkcji.

Funkcja zostanie wykonana przez wywołanie funkcji.

Można wywołać funkcję z dowolnego miejsca na stronie.

Parametry i argumenty
Definiując podprogram programista posługuje się parametrem, czyli informacją (np. wartością), która nie jest znana w momencie definiowania podprogramu, a jedynie zadeklarowaną w jego nagłówku. Argument jest natomiast informacją (np. wartością), znaną, przekazaną z miejsca wywołania, którą posługuje się zdefiniowany wcześniej podprogram.
Parametry-w definicji funkcji;
Argumenty-w wywołaniu funkcji

Nazwa funkcji musi się składać wyłącznie z liter, cyfr i znaków podkreślenia (ponadto nie może zaczynać się cyfrą). Nie można również tworzyć nazw funkcji kolidujących z nazwami wbudowanych funkcji php. Ponadto nie można zdefiniować dwóch funkcji o tej samej nazwie (wielkość liter nie ma znaczenia), nawet jeśli przyjmują różne zestawy parametrów.

Przykłady funkcji w PHP

```php
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>działanie funkcji w PHP- wizytowka</title>
    <meta charset="UTF-8" />
</head>
<body>
  <?php
  //funkcja pobierająca argument i nie zwracająca wartości
  function wizytowka($im, $naz){
   echo '<h3 style="color:red;">*****************************<br>';
   echo '|*|  '.$im.'<br>';
   echo '|*|  '.$naz.'<br>';
    echo '*****************************</h3><br>';
    }
  $imie='Aleksandra';
  $nazwisko='Nowak';
  //wywołujemy funkcję
  wizytowka($imie,$nazwisko);//parametry aktualne
  echo 'dane kolejnej osoby<br>';

  $imie2='Jan';
  $nazwisko2='Kowalski';
   //wywołujemy funkcję
  wizytowka($imie2,$nazwisko2);//parametry aktualne
 ?>
</body>
</html>
```

****
|*| Aleksandra
|*| Nowak
****

dane kolejnej osoby

****
|*| Jan
|*| Kowalski
****

Funkcje w PHP -parametry domyślne

```php
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>działanie funkcji w PHP- suma liczb</title>
    <meta charset="UTF-8" />
</head>
<body>
  <?php
  //funkcja pobierająca argumenty i zwracająca wartość
  //podane wartości domyślne
  function sumaLiczb($a=0, $b=0){
   return $a+2*$b;
    }
  $s1=sumaLiczb(3,6);
  $s2=sumaLiczb(6);
  $s3=sumaLiczb();
  echo 's1='. $s1.'<br>';
  echo 's2='. $s2.'<br>';
  echo 's3='. $s3.'<br>';
 ?>
</body>
</html>
```

Jeśli chcemy, by funkcja miała kilka argumentów domyślnych, to argumenty takie muszą być na końcu listy:

s1=15
s2=6
s3=0

Zasięg zmiennych w PHP

Zasięg zmiennej zależy od miejsca, w jakim ją zdefiniowano. Najczęściej zmienne PHP widoczne są tylko w jednym zasięgu.
Każda zmienna użyta wewnątrz funkcji jest domyślnie ograniczona do zasięgu lokalnego funkcji.

Zmienne globalne są widoczne w całym skrypcie. Są to zmienne, które zostały zdefiniowane poza funkcją. Można z nich korzystać w każdym miejscu skryptu z wyjątkiem wnętrza funkcji. Taki sposób działania zmiennych globalnych jest charakterystyczny dla PHP. W większości języków programowania do zmiennych globalnych można odwołać się z dowolnego miejsca.

```
<body>
<h3>Zasięg zmiennych</h3>
  <?php
 $a=3;//zmienna o zasięgu globalnym
 function wypiszWartosc(){
    echo '<br> wartosc zmiennej wewnątrz funkcji='.$a.'<br>';//zmienna lokalna $a
 }// koniec definicji funkcji
 //wywołujemy funkcję:
wypiszWartosc();

 ?>
</body>
```

Ten skrypt nie wyświetli niczego, ponieważ instrukcja echo odwołuje się do zmiennej lokalnej `$a`, której jak dotąd nie została przypisana żadna wartość. Można tu zauważyć różnicę w stosunku do języka C, gdzie zmienne globalne są zawsze dostępne wewnątrz definicji funkcji, o ile nie zostały nadpisane przez lokalną definicję zmiennej. Może to spowodować problem, że ktoś może nieodwracalnie zmienić wartość zmiennej globalnej. W PHP zmienne globalne muszą być jawnie określone jako globalne wewnątrz funkcji, w której mają być użyte, do czego używamy słowa kluczowego global.

Zasięg zmiennych w PHP-polecenie global

```
<body>
<h3>Zasięg zmiennych-polecenie global</h3>
  <?php
 $a=3;  //zmienna o zasięgu globalnym
 function wypiszWartosc(){
    global $a;
    echo '<br> wartość zmiennej  $a   wewnątrz funkcji  wypiszWartosc jest równa '.$a.'<br>';
 }// koniec definicji funkcji
 //wywołujemy funkcję:
wypiszWartosc();


 function sprawdzGlobal(){
    global $b;
    $b=6;
    echo '<br> wartość zmiennej $b wewnątrz funkcji sprawdzGlobal jest równa '.$b.'<br>';
 }// koniec definicji funkcji
 //wywołujemy funkcję:
sprawdzGlobal();
  echo '<br> wartość zmiennej  $b poza funkcją sprawdzGlobal jest równa '.$b.'<br>';
 ?>
</body>
```

wartość zmiennej $a wewnątrz funkcji wypiszWartosc jest równa 3

wartość zmiennej $b wewnątrz funkcji sprawdzGlobal jest równa 6

wartość zmiennej $b poza funkcją sprawdzGlobal jest równa 6

Funkcje w PHP
-zmienna globalna jako parametr

Przy przekazaniu zmiennej globalnej jako parametru, operujemy na jej kopii, tak jak przy przekazaniu przez wartość w C++.

```
$g = 3;
function f($parametr)
{
var_dump($parametr);
$parametr = 17;
}
f($g);
var_dump($g);
int(3) int(3)
```

Przy przekazaniu zmiennej globalnej za pomocą dyrektywy global operujemy na samej Zmiennej. Efekt podobny, jak przy przekazaniu przez referencję w C++:

```
$g = 3;
function f()
{
global $g;
var_dump($g);
$g = 17;
}
f($g);
var_dump($g);
int(3) int(17)
```

Zmienne statyczne

Zmienne statyczne to zmienne lokalne funkcji, które zachowują swoją wartość między wywołaniami funkcji. Domyślnie zmienna lokalna jest tworzona w chwili wywołania funkcji i jest widoczna w obrębie tej funkcji. Gdy funkcja zakończy działanie, zmienna znika. Przy ponownym wywołaniu funkcji zmienna jest tworzona i przypisywana jest jej wartość początkowa.

```php
<?php
function f1(){
$i=1;
echo '<br> funkcja f1 wywołana' .$i.' raz(y)<br>';
$i++;
}
f1();
 f1();
  f1();
   f1();
function f2(){
static $i=1;
echo '<br> funkcja f2 wywołana ' .$i.' raz(y)<br>';
$i++;
}
f2();
 f2();
  f2();
   f2();
    f2();
?>
```

Przy ponownym wywołaniu funkcji instrukcja przypisująca początkową wartość zmiennej będzie ignorowana, a stosowana będzie wartość zapamiętana przy poprzednim wywołaniu funkcji

funkcja f1 wywołana1 raz(y)

funkcja f1 wywołana1 raz(y)

funkcja f1 wywołana1 raz(y)

funkcja f1 wywołana1 raz(y)

funkcja f2 wywołana 1 raz(y)

funkcja f2 wywołana 2 raz(y)

funkcja f2 wywołana 3 raz(y)

funkcja f2 wywołana 4 raz(y)

funkcja f2 wywołana 5 raz(y)

Funkcje w PHP
-argumenty przekazane przez referencję

Parametry można przekazywać również przez referencję (operator referencji to &). Referencja oznacza, że nie tworzymy lokalnej kopii zmiennej lecz modyfikujemy przekazany argument.

Domyślnie, argumenty funkcji są przekazywane przez wartość (a więc jeśli zmienimy wartość argumentu wewnątrz funkcji, nie zmieni się ona poza funkcją). Jeśli chcemy, aby funkcja zmodyfikowała argumenty, musimy przekazać je przez referencję.

Funkcje mogą ponadto zwracać referencję. Dla określenia, że funkcja powinna zwracać referencję, trzeba umieścić operator referencji przed nazwą funkcji. Niezbędne jest również określenie, że wynik funkcji również powinien być referencyjny.

```php
<?php
 echo '<h3>Argumenty przekazane przez wartośc. Argumenty przekazane przez referencję  </h3>';
 //parametry formalne
 function funkcjaZmien($a, &$b)//pierwszy parametr przekazany przez wartość,
 //drugi-przez referencję
{ $a=2*$a;
$b=2*$b;
 }
$a=3;
$b=3 ;
 //wywołujemy funkcję z parametrami aktualnymi:
 funkcjaZmien($a,$b);
 //wartości zmiennych po działaniu funkcji
 echo '$a='.$a.'-wartość nie została zmieniona<br>';
 echo '$b='.$b.'-wartość została zmieniona<br>';
 ?>
```

Funkcje w PHP
-argumenty przekazane przez referencję

```php
<?php
echo '<h3>Argumenty przekazane przez wartośc. Argumenty przekazane przez referencję  </h3>';
//parametry formalne :$string1, &$string2
function dodaj_cos_extra($string1, &$string2)//pierwszy parametr przekazany przez wartość, drugi-przez referencję
{
   $string1 .= 'Jestem przekazany przez wartość. Funkcja mnie zmieniła. ';
   $string2 .= 'Jestem przekazany przez referencję. Funkcja mnie zmieniła. ';
   echo 'Wartości argumentów przed zakończeniem działania funkcji:<br>';
   echo 'pierwszy argument-->'.$string1.'<br>';
echo 'drugi argument-->'.$string2.'<br>';
}//koniec definicji funkcji
//definiujemy dwie zmienne
$str1='Jestem pierwszym argumentem. ';
$str2='Jestem drugim argumentem.  ';
echo 'Wartosci zmiennych $str1 i $str2 przed działaniem funkcji: <br>';
echo '$str1='.$str1.'<br>';
echo '$str2='.$str2.'<br>';
echo '------------------------------<br>';
echo '<h3>Wywołujemy funkcję.  </h3>';
//wywołujemy funkcję z parametrami aktualnymi-argumentami:$str1, $str2;
dodaj_cos_extra($str1, $str2);
echo '------------------------------<br>';
echo 'Wartosci zmiennych $str1 i $str2 po zakończeniu działania funkcji: <br>';
echo '$str1='.$str1.'<br>';
echo '$str2='.$str2.'<br>';
 ?>
```

Argumenty przekazane przez wartośc. Argumenty przekazane przez referencję

Wartosci zmiennych $str1 i $str2 przed działaniem funkcji:
$str1=Jestem pierwszym argumentem.
$str2=Jestem drugim argumentem.

Wywołujemy funkcję.

Wartości argumentów przed zakończeniem działania funkcji:
pierwszy argument-->Jestem pierwszym argumentem. Jestem przekazany przez wartość. Funkcja mnie zmieniła.
drugi argument-->Jestem drugim argumentem. Jestem przekazany przez referencję. Funkcja mnie zmieniła.

Wartosci zmiennych $str1 i $str2 po zakończeniu działania funkcji:
$str1=Jestem pierwszym argumentem.
$str2=Jestem drugim argumentem. Jestem przekazany przez referencję. Funkcja mnie zmieniła.

Funkcje w PHP -funkcje anonimowe

Od php 5.3 wprowadzono tzw. funkcje anonimowe (lub domknięcia). Anonimowe funkcje mogą być przypisywane bezpośrednio do zmiennej.

```php
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>funkcja anonimowa</title>
    <meta charset="UTF-8" />
</head>
<body>
  <?php
  $podajLogin = function($login)
{
    echo '<h2>Twój login: '.$login.'</h2><br>';
};
$podajLogin('aBcD');

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

Twój login: aBcD

Funkcje w PHP
-funkcje rekurencyjne

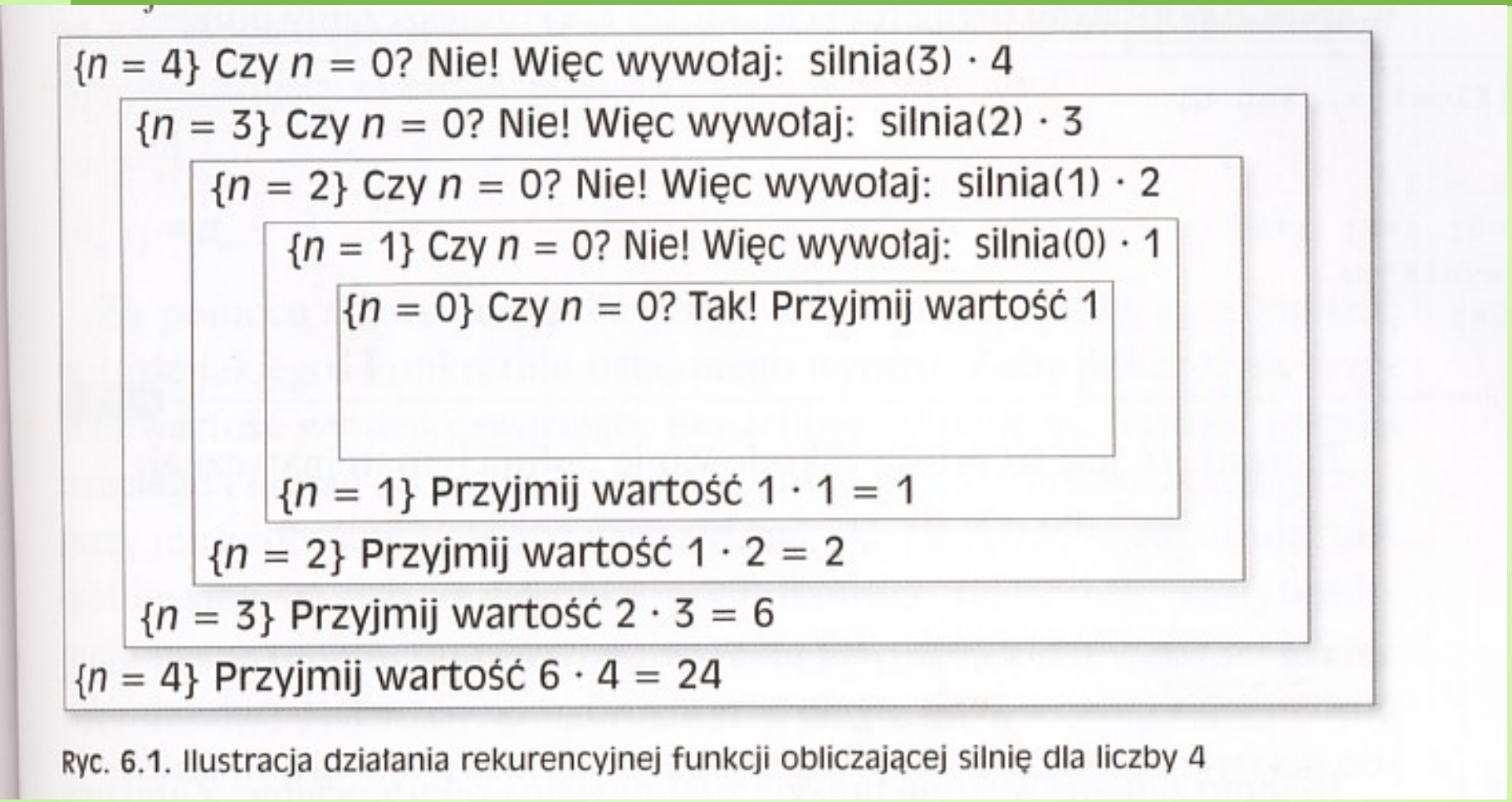
W matematyce funkcją rekurencyjną nazywamy funkcję, która jest zdefiniowana za pomocą samej siebie.
W informatyce funkcją rekurencyjną nazywamy funkcję, która wywołuje samą siebie.

```php
<?php
function  silnia_iteracyjnie($n)
      {
      $silnia=1;
       for ($i=2; $i<=$n; $i++)
       {$silnia=$silnia*$i;}
       return $silnia;
      }

function  silnia_rek($n)
 {if ($n==0) //warunek początkowy rekurencji
       return 1;
      else
      return silnia_rek($n-1)*$n; //rekurencyjne wywołanie funkcji
}

echo silnia_iteracyjnie(4).'<br>';
echo silnia_rek(4).'<br>';
 ?>
```

Funkcje w PHP
-funkcje rekurencyjne



Ryc. 6.1. Ilustracja działania rekurencyjnej funkcji obliczającej silnię dla liczby 4

Każde wywołanie funkcji jest związane z zapamiętaniem kopii wszystkich zmiennych funkcji oraz z umieszczeniem na stosie adresu powrotnego, czyli miejsca w programie, do którego system ma powrócić po zakończeniu funkcji.

Stosem nazywamy obszar pamięci, w którym są przechowywane zmienne wywoływanych funkcji oraz adresy powrotne. Stos charakteryzuje się tym, że jest strukturą powoływaną dynamicznie, to znaczy nie ma stałego miejsca w pamięci i jest powiększany w zależności od potrzeb.

Funkcje w PHP -funkcje rekurencyjne

Plusy i minusy stosowania funkcji rekurencyjnych

Podstawową zaletą funkcji rekurencyjnych jest prostota kodu. Na funkcji obliczającej silnię nie widać tego tak znacząco, lecz pisząc bardziej rozbudowane konstrukcje, jest to zauważalne. Programiści jednak odchodzą od stosowania funkcji rekurencyjnych z racji dużej ilości pamięci, zajmowanej podczas kolejnych wywołań.

Księga gości

Pliki:

dane.php

ksiega.php

Przykłady funkcji wbudowanych:

Tablice – funkcje sortujące

W PHP każda tablica jest tablicą asocjacyjną, to jest składa się z pary „klucz – wartość”. Nawet jeżeli używana jest jako zwykła tablica, znana z innych języków programowania, to i tak jest to tablica asocjacyjna, po prostu klucze są kolejnymi liczbami całkowitymi od 0. Kluczami mogą być liczby całkowite lub łańcuchy tekstowe. W jednej tablicy można mieszać klucze liczbowe i tekstowe. Oczywiście klucze liczbowe nie muszą przebiegać kolejno. Para „klucz – wartość” zapisywana jest w PHP za pomocą operatora „=>”. Oto zatem przykładowa tablica:

```
82 => 'iweifj', 'js7d' => 367, 'dfd' => $jakas_tablica, 73 => $obiekt;
```

O ile kluczem tablicy może być tylko liczba całkowita lub tekst, o tyle wartość może być dowolnego typu (można dowolnie mieszać typy w jednej tablicy). Dzięki tej elastyczności tablica w PHP może być używana jako zwykła tablica, mapa (tablica asocjacyjna), lista i inne struktury. W szczególności elementem tablicy może być inna tablica, w ten sposób w PHP implementujemy tablice wielowymiarowe.

W tablicy PHP należy odróżnić kolejność elementów, od kolejności kluczy. W klasycznej tablicy z C element [3] zawsze występuje przed [4], w PHP element o kluczu 3 wcale nie musi występować przed elementem o kluczu 4. Kolejność elementów w tablicy wynika z kolejności wstawiania, może być zmieniana przez funkcje sortujące.

Przykłady funkcji wbudowanych:

Tablice – funkcje sortujące

Rozpatrywanie zagadnienia sortowania wymaga rozpatrzenia w jaki sposób PHP porównuje ze sobą elementy różnych typów. Operacja porównania dwóch elementów sortowanej tablicy jest podstawą każdego algorytmu sortowania. W klasycznych językach z silną typizacją tablica po prostu zawsze składa się z elementów tego samego typu i problem porównania dwóch elementów zwykle nie nastręcza problemów. W PHP tablica może zawierać wartości różnych typów i przy ich porównywaniu możemy się natknąć na różne niespodzianki. Oto kilka typowych zagwozdek:

- przy porównaniu liczby z tekstem, **tekst jest zamieniany na liczbę**,
- przy porównaniu dwóch tekstów reprezentujących liczby, **są one zamieniane na liczby**,
- porównywanie ze sobą dziwnych rzeczy w rodzaju tablice, zasoby, czy obiekty, nie jest generalnie zalecane, jeżeli ktoś się jednak uprze powinien poczytać dokumentację a potem przetestować efekty na tej wersji PHP, na której będzie docelowo pracował skrypt (mogą być np. różnice między PHP 4 i 5).

PHP posiada nadspodziewanie wiele funkcji do sortowania tablic. Fakt ten wynika, że każdy element tablicy jest parą klucz – wartość, a to stwarza wiele możliwości sortowania.

Przykłady funkcji wbudowanych:

Tablice – funkcje sortujące

Sortowanie tradycyjne – sort() i rsort()

To sortowanie opiera się wyłącznie na wartościach elementów, całkowicie ignorując klucze. Elementy zostają poprzestawiane według narastających (lub malejących dla *rsort()*) wartości, stare klucze zostają usunięte, następnie zostają stworzone nowe klucze liczbowe od 0 zwiększające się o 1 przy każdym elemencie.

$tablica

klucz	wartość
28	-177
71	'jabłko'
19	'pisak'
36	92

sort($tablica)

klucz	wartość
0	-177
1	'jabłko'
2	'pisak'
3	92

rsort($tablica)

klucz	wartość
0	92
1	'pisak'
2	'jabłko'
3	-177

W tym przypadku sortowanie odbywa się w ten sposób, że łańcuch tekstowy przy porównywaniu z liczbą jest konwertowany na liczbę, co przy łańcuchach nie będących reprezentacją liczb zmienia je na 0. Przy porównaniu dwóch łańcuchów natomiast stosuje się porównywanie tekstu (litera po literze). Sposób porównywania można zmienić podając drugi parametr funkcjom sortującym, określający sposób konwersji wartości przed porównaniem.

Przykłady funkcji wbudowanych:

Tablice – funkcje sortujące

Sortowanie tradycyjne – sort() i rsort()

$tablica

klucz	wartość
28	-177
71	'jabłko'
19	'pisak'
36	92

sort($tablica, SORT_NUMERIC)

klucz	wartość
0	-177
1	'pisak'
2	'jabłko'
3	92

rsort($tablica, SORT_NUMERIC)

klucz	wartość
0	92
1	'pisak'
2	'jabłko'
3	-177

Tym razem wszystkie teksty są przed porównaniami konwertowane na liczby. Oczywiście teksty nie reprezentujące liczb dają w wyniku konwersji 0 i ich wzajemna kolejność nie jest określona.

$tablica

klucz	wartość
28	-177
71	'jabłko'
19	'pisak'
36	92

sort($tablica, SORT_STRING)

klucz	wartość
0	-177
1	92
2	'jabłko'
3	'pisak'

rsort($tablica, SORT_STRING)

klucz	wartość
0	'jabłko'
1	'pisak'
2	92
3	-177

Tym razem liczby są przed porównaniem konwertowane na łańcuchy tekstowe. Jeżeli teksty zawierają znaki narodowe należy użyć SORT_LOCALE_STRING.

Przykłady funkcji wbudowanych:

Tablice – funkcje sortujące

Sortowanie z pozostawieniem kluczy – asort() i arsort()

Sortowanie jest również według wartości (zatem kolejność elementów tablicy będzie taka sama jak przy *sort()*), ale zostają pozostawione stare klucze elementów.

$tablica

klucz	wartość
28	-177
71	'jabłko'
19	'pisak'
36	92

asort($tablica)

klucz	wartość
28	-177
71	'jabłko'
19	'pisak'
36	92

arsort($tablica)

klucz	wartość
36	92
19	'pisak'
71	'jabłko'
28	-177

Drugi parametr, definiujący sposób porównywania wartości przy sortowaniu, ma to samo znaczenie jak dla funkcji *sort()*.

Przykłady funkcji wbudowanych:

Tablice – funkcje sortujące

Sortowanie według kluczy – ksort(), krsort()

Tym razem tablica zostaje posortowana według kluczy, oczywiście przemieszczane są całe elementy, a więc klucz z odpowiadającą mu wartością.

$tablica

klucz	wartość
28	-177
71	'jabłko'
19	'pisak'
36	92

ksort($tablica)

klucz	wartość
19	'pisak'
28	-177
36	92
71	'jabłko'

krsort($tablica)

klucz	wartość
71	'jabłko'
36	92
28	-177
19	'pisak'

Również w tym przypadku sposób sortowania **kluczy** można zmienić podając drugi parametr funkcji sortującej.

Generalnie wymieszanie różnych typów wartości (dla *sort()*, *rsort()*, *asort()*, *arsort()*), czy różnych typów kluczy (dla *ksort()* i *krsort()*) jest **złym pomysłem** i powinno się tego unikać aby zapobiec problemom z nieoczekiwanymi wynikami sortowania. Jeżeli programista decyduje się na sortowanie wymieszanych typów, powinien mieć pełną świadomość tego co robi.

Przykłady funkcji wbudowanych:

Tablice – funkcje wyszukiwania

klucz array-search(poszukiwana_wartość, $tablica);

Funkcja zwraca index pierwszego wystąpienia elementu

d

```php
<?php
$tab=array("a"=>"blue","b"=>"green","c"=>"blue","d"=>"red", "e"=>"red" );
echo array_search("red", $tab);
echo '<br>';
echo array_search("yellow", $tab);
?>
```

Przykłady funkcji wbudowanych:

Funkcje losowe

Liczby losowe mają szerokie zastosowanie w programowaniu serwisów www. Przydają się przy generowaniu losowych haseł, tymczasowych kodów weryfikujących lub linków aktywacyjnych. Wyświetlanie różnych cytatów, czy losowych produktów w sklepie również opiera się o liczby losowe.

Przykłady funkcji wbudowanych:

Funkcje losowe

int rand ([int $min [, int $max]])-

Funkcja wywołana bez opcjonalnych argumentów min i max, funkcja rand() zwraca pseudolosową liczbę ze zbiory {0, 1, 2, ... RAND_MAX}. Dla uzyskania liczby losowej z przedziału np. od 5 do 15 (włącznie), należy wywołać rand (5,15).
Funkcja getrandmax() zwraca maksymalną wartość zwracaną przez funkcję rand().

Funkcją do generowania liczb losowych, działającą według szybszego algorytmu(średnio cztery razy szybszego) jest funkcja mt_rand(). -Korzysta ona z generatora liczb losowych Mersenne Twister, który generuje liczby losowe wystarczające do inicjowania niektórych rodzajów funkcji kryptograficznych.

Przykłady funkcji wbudowanych:

Funkcje losowe

```
8   <?php
9   // Jeżeli wersja PHP < 4.2.0 wtedy zachodzi potrzeba użycia srand()
10  // srand(floor(time() / (60*60*24)));
11  for($i=0;$i<=10; $i++){
12  $tab[$i]=rand(0,20)-10;
13  }
14  foreach($tab as $klucz=>$wartosc){
15    echo $klucz.' => '.$wartosc.'<br>';
16  }
17   ?>
18  </body>
19  </html>
```

```
0 => -7
1 => 9
2 => 3
3 => -5
4 => 4
5 => -3
6 => 4
7 => 6
8 => 2
9 => -6
10 => 0
```

Przykłady funkcji wbudowanych:

Funkcje losowe – funkcja shuffle($tablica)

```
<body>
<h3>losowe ustawienie elementów tablicy, funkcja suffle</h3>
 <?php
$my_array = array("a"=>"red","b"=>"green","c"=>"blue","d"=>"yellow","e"=>"purple");

shuffle($my_array);
print_r($my_array);
?>

<p>odśwież stronę</p>

</body>
```

losowe ustawienie elementów tablicy, funkcja suffle

Array ([0] => red [1] => blue [2] => purple [3] => green [4] => yellow)

odśwież stronę

Przykłady funkcji wbudowanych:

Za każdym przeładowaniem strony chcemy pokazać na niej inną, losową poradę (cytat lub inną informację) wybraną z pewnej, przygotowanej wcześniej puli porad.

Formularze w PHP

Po kliknięciu przycisku „wyślij”, dane z formularza powinny zostać przesłane na serwer za pomoca jednej z dwóch metod: POST oraz GET. Metoda post jest używana, kiedy parametrów jest niewiele. W tej metodzie parametry są przekazane za pomocą adresu URL. Schematyczna postać URL wygląda następująco:

Http://www.nazwaserwera.pl/strona.php?parametr1=wartosc1&parametr2=wartosc2

Długość adresu URL jest ograniczona, a przekazane parametry są widoczne w pasku adresu przeglądarki, dlatego metoda ta jest stosowana do przesyłanie niedużych ilości danych.

Metoda POST do przekazywania parametrów wykorzystuje nagłówek strony. Metoda ta umożliwia przekazywanie większej ilości parametrów, a parametry nie są widoczne w pasku przeglądarki.

$_GET[], $_POST[] - tablice superglobalne – widoczne w każdym miejscu kodu PHP

Formularze w PHP

```
<h3>formularz</h3>
<form action="skrypt_form.php" method="POST">
imie: <input type="text" name="imie"><br>
nazwisko: imie: <input type="text" name="nazwisko"><br>
wykształcenie: <br> <input type="radio" value="podstawowe" name="wykształcenie" checked>podstawowe<br>
  <input type="radio" value="srednie" name="wykształcenie" checked>srednie<br>
   <input type="radio" value="wyzsze" name="wykształcenie" checked>wyższe<br>
   <input type="checkbox"  name="opcja" maxlength="1">zgadzam się na przetwarzanie moich danych osobowych<br>

   <input type="submit" value="Wyślij" name="wyslij" >
   <input type="reset" value="Wyczyść" name="resetuj" >
   </form>
</form>
```

```
<?php
echo 'odpowiedź z PHP  <br>';
echo $_POST['imie'].'<br>';
echo $_POST['nazwisko'].'<br>';
echo $_POST['wykształcenie'].'<br>';
?>
```

formularz

imie:

nazwisko: imie:

wykształcenie:

- ○ podstawowe
- ○ srednie
- ◉ wyższe

☐ zgadzam się na przetwarzanie moich danych osobowych

Wyślij | Wyczyść

Formularze w PHP

Funkcja isSet($zmienna)- pozwala stwierdzić, do skryptu została przekazana wartość danego pola.

Funkcja empty($zmienna) sprawdza, czy istniejąca zmienna użyta jako argument jest pusta

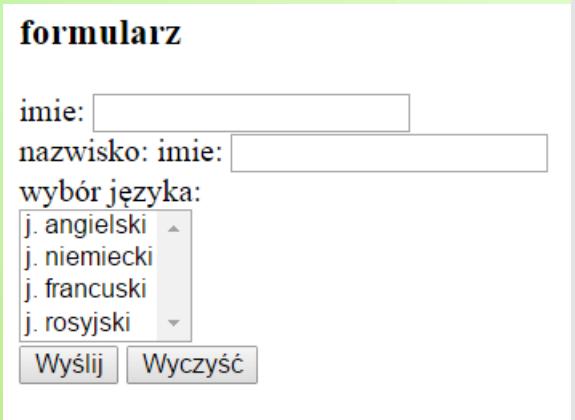
```php
<?php
echo 'odpowiedź z PHP  <br>';
echo $_POST['imie'].'<br>';
echo $_POST['nazwisko'].'<br>';
if (empty($_POST['nazwisko'])){
echo '<br>należy wypełnić pole nazwisko';}
   else {echo $_POST['nazwisko'].'<br>';}
echo '<br>wyksztalcenie'.$_POST['wyksztalcenie'].'<br>';
if(isset($_POST['opcja'])) {echo 'wybrano zgode na przetwarzanie danych osobowych <br>';}
?>
```

Formularze w PHP-pole typu select

W formularzu może wystąpic pole typu SELECT, które służy do wyświetlania listy wartości i pozwala na wybranie jednej z nich lub kilku.

Tu nawiasy kwadratowe!
Elementy wybrane w polu „języki[]” będą dostępne w tablicy $_POST['jezyki']

```
<body>
<h3>formularz</h3>
<form action="skrypt_do_SELECT.php" method="POST">
imie: <input type="text" name="imie"><br>
nazwisko: <input type="text" name="nazwisko"><br>
wybór języka:<br>
<select name="jezyki[]" multiple> <br><br>
        <option value="angielski">j. angielski </option>
        <option value="niemiecki">j. niemiecki </option>
        <option value="francuski">j. francuski </option>
        <option value="rosyjski">j. rosyjski </option>
        </select>
        <br>
    <input type="submit" value="Wyślij" name="wyslij" >
    <input type="reset" value="Wyczyść" name="resetuj" >
    </form>
</form>
</body>
```



Formularze w PHP-pole typu select

W formularzu może wystąpić pole typu SELECT, które służy do wyświetlania listy wartości i pozwala na wybranie jednej z nich lub kilku.

Funkcja if (!empty($_POST['jezyki'])) sprawdza, czy tablica $_POST['jezyki'] jest pusta

```php
<?php
echo 'dane z formularza:  <br>';

if (!empty($_POST['jezyki'])){
      echo $_POST['imie'].'<br>';
echo $_POST['nazwisko'].'<br>zna:';
echo '<ul>';
foreach($_POST['jezyki'] as $w)
  echo '<li>'.$w.'</li>';
  echo '<ul>';}

  else {echo $_POST['nazwisko'].'nie zna żadnego jezyka';}
?>
```

dane z formularza:
Alicja
Nowak
zna:

- angielski
- francuski

Formularze w PHP-przesyłanie plików na serwer

<form enctype="multipart/form-data" action="formularz_plik_na_server.php" method="POST">

enctype – określa sposób kodowania danych

Tablica $_FILES zawiera dane pliku wysłanego przez formularz HTML.

$_FILES['plik']['name'] **Oryginalna nazwa wysyłanego pliku.**

$_FILES['plik']['type'] **Typ MIME wysyłanego pliku (JPEG, GIF, ...).**

$_FILES['plik']['size'] **Rozmiar wysyłanego pliku (w bajtach)**

$_FILES['plik']['tmp_name'] **Tymczasowa nazwa pliku, który został wysłany na serwer. Wysłany do serwera plik jest umieszczony w katalogu tymczasowym. Do przeniesienia pliku do właściwej lokalizacji służy finkcja:**

move_uploaded_file($_FILES['plik']['tmp_name'],$nowa_nazwa);

$_FILES['plik']['error'] **Numer błędu (0 oznacza prawidłowe wysłanie).**

Formularze w PHP-przesyłanie plików na serwer

Przed przesłaniem pliku do katalogu docelowego można sprawdzić za pomocą funkcji is_uploaded_file($_FILES['plik']['tmp_name']), czy plik tymczasowy istnieje.
Aby plik mógł zostać przeniesiony w docelowe miejsce, użytkownik musi mieć odpowiednie prawa dostępu do katalogu docelowego

```
<body>
<h3>formularz-przesyłanie plików</h3>
<form enctype="multipart/form-data" action="formularz_plik_na_server.php" method="POST">
 <input type="hidden" name="max_file_size" value="50000">
 <p>wyslij plik:
 <input type="file" name="plik" size="30">
  <input type="submit" value="Wyślij" name="wyslij" ></p>
</form>
<?php
$katalog_plik="./"; //katalog docelowy
$max_rozm=$_POST['max_file_size'];
if(is_uploaded_file($_FILES['plik']['tmp_name'])){
if($_FILES['plik']['size']>$max_rozm)
                {echo 'plik za duży';}
                    else {
                    echo 'plik '.$_FILES['plik']['name'].'zostal przeslany<br>';
                    if (isset($_FILES['plik']['type']))
                            echo 'typ pliku'.$_FILES['plik']['type'].'<br>';
                            $nazwa=$katalog_plik.$_FILES['plik']['name'];
                            move_uploaded_file($_FILES['plik']['tmp_name'],$nazwa);
                    }
}
else {echo 'bład podczas przesylaniea pliku';}
?>
</body>
```

Pliki cookies w PHP

Metodą nie tyle przekazywania parametrów, co przechowywania niewielkich ilości danych na komputerze oglądającego stronę (np. informacje o imieniu i nazwisku lub nazwie użytkownika w tym serwisie) jest mechanizm cookies (ciasteczka). Ciasteczka ustawione przez dany serwis dostępne są tylko dla niego i ustawiane są na jakiś czas.

Ciasteczka przekazywane są za pomocą nagłówków HTTP. Muszą być one wysłane zanim do przeglądarki zostanie wysłana jakakolwiek inna treść. W związku z tym przed zapisaniem ciasteczka nie może być żadnego wywołania funkcji echo i pochodnych, a także tag otwierający tryb PHP musi być pierwszymi znakiami w pliku - nie może być żadnej spacji ani pustych wierszy.

Zasada działania plików cookies:

1) po nawiązaniu połączenia server wysyła do przeglądarki nagłówek Set-Cookie, który zawiera plik cookie

2) przeglądarka zapisuje plik cookie na dysku użytkownika

3) przy kolejnym połączeniu z serwerem przeglądarka wysyła do niego przechowywany na dysku plik cookie

Pliki cookies w PHP

PHP automatycznie odczytuje ciasteczka i zamienia je na zmienne. Są one przechowywane w superglobalnej tablicy asocjacyjnej $_COOKIE (dawniej $HTTP_COOKIE_VARS), w której kluczami są nazwy ciasteczek. Ciasteczka ustawia się je pomocą funkcji **setcookie(nazwa, wartość, czas_wygaśnięcia, ścieżka, domena, bezpieczeństwo)**. Tylko pierwszy parametr jest niezbędny. Oznacza on nazwę cookiesa - taką nazwą będzie miała zmienna stworzona przez PHP po ponownym odczytaniu ciastek. Funkcja parametru "wartość" - taka wartość będzie przechowana w ciasteczku o podanej nazwie. Parametr czas_wygaśnięcia oznacza czas, po jakim ciastko zostanie skasowane. Czas ten należy podać jako ilość sekund od 1.1.1970 - tak jest przechowywany czas w systemach UNIX'owych. Aktualny czas w tym formacie zwracany jest przez funkcję **time()**. Jeśli cookie ma być trzymany przez godzinę, to do czasu zwróconego przez time() należy dodać ilość sekund zawartych w godzinie - "time() + 3600". Podobnie należy postępować w przypadku innych przedziałów czasu:

- godzina - time()+3600
- dzień - time()+86400

Jeżeli nie został podany czas wygaśnięcia ciasteczka, to plik traci swoją ważnośc po zamknięciu przeglądarki.

Pliki cookies w PHP

Każdy wysłany plik cookie jest automatycznie umieszczany w superglobalnej tablicy $_COOKIE. Jej indeksami są nazwy plików cookies. Dostęp do pliku cookie jest możliwy przez:
$_COOKIE['nazwa_pliku']

setcookie('nazwa') -'nazwa' jest nazwą pliku cookie.

skrypt tworzący cookie

```php
<?php
setcookie("wizyta", date("Y-m-d, G:i:s"),2592000+time());
//format daty: G- godzina i - minuty, s-sekundy
//skrypt tworzący plik cookie
//termin wygaśnięcia to 30 dni od chwili utworzenia: 2592000=60s*24godz*30dni
?>
```

Drugi skrypt odczytujący cookie

```php
<?php
if (isset($_COOKIE['wizyta'])){
$ostatnia=$_COOKIE['wizyta'];
echo 'witamy ponownie! Ostatnio odwiedziles nas  '.$ostatnia;
}
?>
```

Pliki cookies w PHP-przekazywanie danych użytkownika

```php
<?php
if(!isset($_COOKIE['dane']) && !isset($_POST['imie'])){
?>
<!DOCTYPE html>
<html lang="pl">
  <head>
    <title>formularz</title>
    <meta charset="UTF-8" />
</head>
<body>
<h3>formularz</h3>
<form action="cookie_przesylanie_danych_uzytkownika.php" method="POST">
Podaj imie: <input type="text" name="imie"><br>
   <input type="submit" value="Wyślij" name="wyslij" >
  </form>
  <?php
  }
  else{
  if( isset($_POST['imie'])){
  setcookie('dane',$_POST['imie'], time()+60);
  echo 'dziękujemy za wprowadzenie danych';
  }
  else {
  echo 'witamy po raz kolejny na naszej stronie'.$_POST['imie'];
  }
  }
  ?>
  </body>
  </html>
```

Pliki cookies w PHP-licznik odwiedzin

```php
<?php
if(!isset($_COOKIE['odwiedz'])) {
$liczba=1;}

else {
 $liczba=intval($_COOKIE['odwiedz'])+1;
}
    setcookie("odwiedz", $liczba, time()+60*60*24*365);
?>
  <!DOCTYPE html>
<html lang="pl">
  <head>
    <title>formularz</title>
    <meta charset="UTF-8" />
</head>
<body>
<?php
$wyraz=(($liczba==1)?"raz":"razy");
echo 'w ciagu ostatniego roku odwiedziles nas '.$liczba.$wyraz;

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