Ćwiczenie 6 - PHP (2) - Tablice w PHP

Tablice są typem zmiennych zawierającym uporządkowany zbiór danych. Pełnią ważną rolę w aplikacjach internetowych z bazą danych – służą do gromadzenia wyselekcjonowanych danych i manipulowania nimi.

Do zmiennych tych uzyskuje się dostęp przez indeks w nawiasie kwadratowym podane bezpośrednio po nazwie zmiennej tablicowej. Podobnie przypisuje się wartość do elementu tablicy. Przykład:

```
<?php
$tablica[0] = "Wpis numer 0";
$tablice[1] = "Wpis numer 1";
$tablica[2] = "Wpis numer 2";
echo $tablica[2]; // Wyświetlony zostanie napis "Wpis numer 2";
?>
```

Elementem tablicy może być każdy typ zmiennej (z innymi tablicami i obiektami włącznie).

Tablica dwuwymiarowa, zawiera dwa indeksy, np.

```
$tablica[0][2] = 1;
```

Tablica asocjacyjna

W PHP występuje też inny rodzaj tablic, tak zwane tablice asocjacyjne (zwane też czasem haszami - ***hash table***). Są to tablice, w których zamiast indeksów liczbowych używa się identyfikatorów znakowych (kluczy):

```
<?php
$tablica["imie"] = "Jan";
$tablica["nazwisko"] = "Kowalski";
$tablica["adres"] = "Polna 1";
echo $tablica["imie"]." ".$tablica["nazwisko"].
        ", ul. ".$tablica['adres']."<BR>";
?>
```

Wymaganie identyczności typu wszystkich elementów tablicy nie jest tu już obowiązujące, działa poniższy skrypt:

```
<?php
$tablica[0]["nazwa"] = "konserwa";
$tablica[0]["cena"] = 10.5;
$tablica[1]["nazwa"] = "chleb";
$tablica[1]["cena"] = 2.76;

for ($x=0;$x<2;$x++)
        {
        echo $tablica[$x]["nazwa"].
                " ".$tablica[$x]["cena"]."<br>";
        };
echo $tablica[0]["nazwa"]+$tablica[1]["cena"];
?>
```

Przy okazji widzimy działanie instrukcji "pętli" liczonej for.

Inny sposób definiowania tablic

Tablice można również definiować wykorzystując składnię **array**() - używaną do tekstowej reprezentacji tablic (nie jest zwykłą funkcją).

Składnia "indeks => wartości", oddzielona przecinkami, definiuje pary indeksów i wartości. Indeks może być łańcuchem lub liczbą. Jeśli indeks zostanie pominięty, automatycznie wygenerowany zostanie indeks będący liczbą całkowitą, poczynając od 0. Jeśli indeks jest liczbą całkowitą, następny wygenerowany indeks będzie miał wartość "największy indeks + 1". Jeśli pojawią się dwie wartości o tym samym indeksie, ostatnia nadpisze wcześniejsze.

Poniższy przykład demonstruje jak stworzyć wielowymiarową tablicę, jak określić klucze w tablicy asocjacyjnej i jak pominąć i kontynuować liczbowe indeksy w normalnych tablicach.

Przykład bardziej złożonego użycia **array**( ):

```
<?php
$owoce = array (
  "owoce"  => array ("a"=>"pomarańcza", "b"=>"banan", "c"=>"jabłko"),
  "liczby" => array (1, 2, 3, 4, 5, 6),
  "dziury" => array ("pierwszy", 5 =>"drugi", "trzeci")
);
echo $owoce["owoce"]["a"]." ".$owoce["owoce"]["b"];
?>
```

Aby **w całości wyświetlić** tablicę (dla celów kontrolnych) – wykorzystujemy funkcje:

print_r ($zmienna_tablicowa);

lub

var_dump($zmienna_tablicowa);

Aby dodać kolejny wpis na końcu tabeli wystarczy przy przypisywaniu wartości nie wpisywać indeksu do nawiasów kwadratowych. Jeśli w ten sposób dodawane są wpisy do nowej tablicy, to pierwszy wpis ma indeks 0.

Indeks można też podawać ze zmiennej, a nawet z innej tablicy czy funkcji:

```
<?php
$tab1[] = 1;
$tab1[] = 0;
$tab1[] = 3;
$tab1[] = 2;

$tab2[] = "Pierwszy";
$tab2[] = "Drugi";
$tab2[] = "Trzeci";
$tab2[] = "Czwarty";

echo $tab2[$tab1[2]];
?>
```

Przeglądanie tablic

Często zachodzi potrzeba wykonania jakiejś operacji na wszystkich elementach tablicy. Sprawa jest prosta jeśli tablica jest zwykłą tablicą z indeksami liczbowymi i znamy liczbę tych elementów:

```
<?php
$tbl[] = 1;
$tbl[] = 2;
$tbl[] = 3;
$tbl[] = 4;
$tbl[] = 5;
for( $x = 0; $x < 5; $x++ ) {
        // Pętla wykona się 5 razy (0...4)
        echo $tbl[$x];
}
?>
```

Gdy nie znamy liczby elementów tablicy korzystamy z funkcji:

count($nazwa_tablicy)

Zwraca ona liczbę elementów w tablicy podanej jako parametr.

```
<?php
for ($x = 0; $x < 5; $x++ )
{
  $tbl[]=$x;
}
for( $x = 0; $x < count($tbl); $x++ ){
  echo $tbl[$x];
}
```

```
?>
```

Jeszcze trudniej jest jeśli konieczne jest przejrzenie tablicy asocjacyjnej. W tym przypadku należy skorzystać z funkcji **list()** i **each()**.

list() - Przypisz zmienne tak jakby były tablicą

Podobnie jak array(), nie jest naprawdę funkcją, ale elementem składni języka. Instrukcja list() jest używana do przypisywania listy zmiennych w jednej operacji.

Uwaga: list() działa tylko z tablicami o indeksach liczbowych, zakładając że indeksy zaczynają się od 0.

Przykład użycia list()

```
<?php
$info = array('kawa', 'brązowa', 'kofeina');
// Listowanie wszystkich zmiennych
list($napój, $kolor, $składnik) = $info;
print $napój." jest ".$kolor." a ".$składnik." czyni ją wyjątkową"."<BR/>";
// Listowanie niektórych elementów
list($napój, , $składnik) = $info;
print $napój." zawiera:". $składnik."<BR/>";
// Albo przeskoczmy od razu do trzeciego
list( , , $składnik) = $info;
echo "Potrzebna jest mi". $składnik."<BR/>";
?>
```

each()

Zwraca bieżącą parę: klucz i wartość z tablicy i przesuwa wewnętrzny wskaźnik tablicy do przodu o jeden element. Para ta jest zwracana jako czteroelementowa tablica, z kluczami **0**, **1**, **key** i **value**. Elementy **0** i **key** zawierają nazwę klucza elementu tablicy, a **1** i **value** zawierają wartość elementu tablicy. Jeśli wewnętrzny wskaźnik tablicy wskazuje na miejsce poza końcem zawartości tablicy, each() zwraca *FALSE*.

Przykłady użycia each()

```
<?php
$imie = array ("Adam", "Barbara", "Cezary", "Damian", "Edward", "Felek");
$wiersz = each ($imie);
?>
```

$**wiersz** zawiera teraz następujące pary klucz/wartość:

```
0 => 0
1 => 'Adam'
key => 0
value => 'Adam'
```

Each() jest zazwyczaj używana w połączeniu z ***list()*** aby "przejść" przez tablicę. Wykorzystuje się tu pętlę warunkową *while*:

```
<?php
$tablica["imie"] = "Jan";
$tablica["nazwisko"] = "Kowalski";
$tablica["adres"] = "Polna 1";
while(list($klucz, $wartosc)=each($tablica))
        echo "klucz=".$klucz." oraz wartość".$wartosc<BR>";
?>
```

Instrukcja(-e) wykonawcza pętli (iteracji) warunkowej *while* powtarzane są dopóki warunek jest prawdziwy – tu warunkiem jest prawidłowe przepisanie do zmiennych kolejnych wartości z tablicy. Jak widać, w każdej iteracji mamy dostępne 2 zmienne, przyjmujące wartości kolejnych kluczy i wartości przypisanych tym kluczom.

Sortowanie tablic

PHP ofertuje cały zestaw funkcji służących do sortowania tablic. Są to: asort(), **arsort**(), **ksort**(), **rsort**(), **sort**(), **uasort**(), **usort**(), i **uksort**(). Większość funkcji (oprócz trzech ostatnich) przyjmuje jeden parametr: zmienną zawierającą tablicę do posortowania. Żadna z funkcji nie zwraca żadnego wyniku. Działanie poszczególnych funkcji:

asort() - sortuje tablice asocjacyjne zachowując przypisanie kluczy do wartości:

```
<?php
$owoce = array ("d"=>"mango", "a"=>"papaja", "b"=>"banan", "c"=>"aronia");
asort ($owoce);
reset ($owoce);
while (list ($klucz, $wartosc) = each ($owoce))
        {echo $klucz."=".$wartosc."<BR>";}
?>
```

Wynikiem działania powyższego przykładu powinno być:

```
c = aronia
b = banan
d = mango
a = papaja
```

- arsort() - sortuje w odwrotnej kolejności tablice asocjacyjne zachowując przypisanie kluczy do wartości. Funkcja prawie identyczna jak poprzednia, tyle że dane sortowane są "od tyłu".
- ksort() - sortuje tablice asocjacyjne według kluczy. Powyższy przykład po podmianie funkcji asort na ksort powinna dać taki wynik:

```
a = papaja
b = banan
c = aronia
d = mango
```

rsort() - sortuje zwykłe tablice (nie asocjacyjne) w odwróconej kolejności

sort() - sortuje zwykłe tablice (nie asocjacyjne) w kolejności alfabetycznej

uasort() - funkcja sortująca tablice asocjacyjne za pomocą zdefiniowanej przez użytkownika funkcji porównującej elementy (nazwa funkcji jest podawana za pomocą drugiego parametru)

usort() - funkcja sortująca zwykłe tablice za pomocą funkcji zdefiniowanej przez użytkownika

uksort() - funkcja sortująca tablice asocjacyjne według klucza za pomocą funkcji zdefiniowanej przez użytkownika.

W trzech ostatnich funkcjach sortujących trzeba jako drugi parametr podać funkcję porównującą elementy tablicy. Jak definiuje się funkcje opisane jest w jednym z następnych rozdziałów. Funkcje takie pobierają 2 argumenty. Zwracane jest 0 jeśli argumenty są sobie równe, -1 jeśli pierwszy argument jest mniejszy od drugiego a 1 jeśli jest większy. Wypróbować inne funkcje sortujące.

Tworzenie ciągów tekstowych z tablic i odwrotnie

PHP umożliwia zamianę ciągów tekstowych na tablice i odwrotnie. Zamiana ciągu na tablicę jest bardzo przydatna jeśli zachodzi potrzeba „wyciągnięcia" jakiegoś fragmentu danych z ciągu. Załóżmy że w odczytaliśmy z pliku z danymi (o odczycie z plików w jednym z kolejnych rozdziałów) linię z logu zapisanego przez licznik WWW: "12/11/2000;19:23:33;Netscape Navigator;192.168.1.1". Jak widać dane rozdzielone są średnikami. Do rozdzielania ciągów na tablicę służy funkcja **explode**(). Jako pierwszy parametr trzeba do niej podać znak lub dłuższy ciąg który oddziela kolejne pola, jako drugi ciąg do rozdzielenia. Opcjonalnie można podać trzeci argument, który oznacza maksymalną liczbę pól - jeśli jest ich więcej niż ta liczba, to ostatnie pole będzie zawierało wszystkie pozostałe pola. Funkcja zwraca tablicę zawierającą kolejne pola.

Przykładowo:

```
<?php
$dane="alfa;beta;gamma;delta";
$tablica = explode(";", $dane);
?>
```

Czasem potrzebne jest działanie w drugą stronę: złącznie pól tablicy w jeden ciąg, w którym pola oddzielone są jakimś znakiem (lub kilkoma). Do tego służy funkcja **implode()**. Jako pierwszy parametr podawany jest ciąg za pomocą którego "sklejane" są elementy tablicy, a jako drugi właśnie tablica do posklejania. Zwracany jest ciąg zawierający "posklejane" elementy. Przykład zastosowania:

```
<?php
$dane = implode(";", $tablica);
?>
```

Zadania

1. Utworzyć dowolną tablicę asocjacyjną, posortować ją według wartości i wyświetlić elementy kolejno na ekranie.
2. Utworzyć tablicę zawierającą kąty i kolejne wartości funkcji sin(x) w zakresie od 0 do 900 z krokiem 50 wyświetlić jej zawartość na ekranie: w 2 kolumnach (kąt - wartość).
3. Postać pętli liczonej:

```
for ($x=0; $x<=18; $x++)
                    {
                    ......
                    }
```

4. Wykonać przykłady i zadania z ćwiczenia i umieścić je na własnym projekcie strony internetowej – wyposażając w hiperłącza do kolejnych zadań (patrz: plik dodatek *HTML*)