Lekcja 45 (kl. II. PR)

Temat: Tablice, czyli jak sortować dane w języku PHP.

Cele lekcji:

– użycie tablic w języku PHP do sortowania danych

Uczeń:

– inicjuje tablice w języku PHP, w tym asocjacyjną;
– sortuje tablicę jednowymiarową i wielowymiarową w PHP.

Podręcznik str. 271 – Informatyka. 2. PR, Operon

Przebieg lekcji:

1. Zapoznanie się z celami lekcji.
2. Pojęcie tablicy w PHP.
3. Inicjowanie tablicy - rys. 44.1 I 44.2
4. Sortowanie tablic.- rys. 44.3 I 44.4
5. Ćwiczenia praktyczne – stosowanie funkcji w PHP.

Podręcznik str. 274.

Materiał:

Wstęp

Tablice są bardzo specyficznym typem zmiennych – są to, najprościej mówiąc, zmienne zawierające w sobie uporządkowany zbiór zmiennych. Do zmiennych tych uzyskuje się dostęp przez liczbę w nawiasie kwadratowym podane bezpośrednio po nazwie zmiennej – tablicy. Liczba ta to tak zwany indeks – numer kolejny zmiennej w tablicy. Tak samo przypisuje się wartość do tablicy.

Przykład 1. Tworzenie tablicy

```
<?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";

?>
```

Aby poprostu 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, z innej tablicy czy funkcji – z dowolnego wyrażenia zwracającego wartość.

Przykład 2. Indeksy tablic

```
<?php

$tab1[] = 1;
$tab1[] = 0;
$tab1[] = 3;
$tab1[] = 2;

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

echo $tab2[$tab1[2]];

?>
```

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

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):

Przykład 3. Tablice asocjacyjne

```
<?php

$tablica["imie"] = "Jan";
$tablica["nazwisko"] = "Kowalski";
$tablica["adres"] = "Polna 1";

echo $tablica["imie"]." ".$tablica["nazwisko"].", ul. ".$tablica["adres"]."n";

?>
```

Przeglądanie tablic

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

Przykład 4. Przeglądanie tablic pętlą

```
<?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];

}

?>
```

Sprawa się trochę komplikuje jeśli nie znamy ilości elementów tablicy. Wtedy z pomocą przychodzi funkcja `count( $nazwa_tablicy )`. Zwraca ona ilość elementów w tablicy podanej jako parametr.

Przykład 5. Pętla for i funkcja count

```
<?php

$tbl[] = 1;
$tbl[] = 2;
$tbl[] = 3;
$tbl[] = 4;
$tbl[] = 5;

for( $x = 0, $cnt = count($tbl); $x < $cnt; $x++ ){

   echo $tbl[$x];

}

?>
```

Jeszcze trudniej jest jeśli konieczne jest przejrzenie tablicy asocjacyjnej, ale i to da się załatwić. W tym przypadku należy skorzystać z funkcji `list()` i `each()`. Nie będę omawiał ich działania – jesli kogoś to interesuje, to odsyłam do manuala PHP. Przy przechodzeniu przez tablice asocjacyjne trzeba wykorzystać pętlę while.

Przykład 6. Przeglądanie tablic asocjacyjnych

```
<?php

$tablica["imie"] = "Jan";
$tablica["nazwisko"] = "Kowalski";
$tablica["adres"] = "Polna 1";

while( list($klucz, $wartosc) = each($tablica) )
   echo "$klucz => $wartosc<BR>";

?>
```

Jak widać, w każdej iteracji pętli 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()`
- `uksort()`

Większość funkcji (oprócz trzech ostatnich) przyjmuje jeden parametr: zmienną zawierającą tablicę do posortowania. Żadna z funkcji nie zwraca żadnego wyniku.

Funkcje sortujące

`asort()`

sortuje tablice asocjacyjne zachowując przypisanie kluczy do wartości

```
<?php

$owoce = array ("d"=>"mango", "a"=>"papaja", "b"=>"banan", "c"=>"aronia");
asort ($owoce);
reset ($owoce); // Funkcja ta powoduje powrót do pierwszego elementu tablicy
while (list ($klucz, $wartosc) = each ($owoce)) {
    echo "$klucz = $wartoscn";
}

?>
```

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 poniższy 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.

Tworzenie ciągów z tablic i odwrotnie

PHP umożliwia zamianę ciągów na tablice i odwrotnie. Zamiana ciągu na tablicę jest bardzo przydatna jeśli zachodzi potrzeba wyciągnięcie 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.

Przykład 7. Rozdzielenie danych separowanych średnikiem

```
<?php

$dane = "12/11/2000;19:23:33;Netscape Navigator;192.168.1.1";
$tablica = explode(";", $dane);

?>
```

Jest także rozszerzona wersja funkcji explode: `split()`. Różni się ona tym, że zamiast prostego ciągu znaków rozdzielających pola, akceptuje ona wyrażenia regularne (co to jest wyrażenie regularne mniej więcej wyjaśniono w rozdziale dotyczącyn ciągów).

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. Jako przykład zastosowania może posłużyć właśnie zapisywanie danych o użytkowniku w aplikacji licznika odwiedzin – tablica zawiera dane o odwiedzającym, a potrzebny jest ciąg pooddzielany średnikami.

Przykład 8. Łączenie danych z tablicy

```
<?php

$dane = implode(";", $tablica);

?>
```