**Лабораторна робота № 14**

**Тема: МАСИВИ У PHP. Частина 2.**

**Мета:** Ознайомитися з прийомами застосування масивів при розв’язуванні задач.

**Асоціативні масиви в PHP**

В PHP індексом масиву може бути не тільки число, а й рядок. Причому на такий рядок не накладаються ніякі обмежень: він може містити пробіли, довжина такого рядка може бути будь-яка.

Асоціативні масиви особливо зручні в ситуаціях, коли елементи масиву зручніше пов'язувати зі словами, а не з числами.

Отже, масиви, індексами яких є рядки, називаються асоціативними масивами.

У наступному прикладові символьні назви слугують як ключі, а числа – як значення.

<?php

$arr = array('the\_first'=>1,'the\_second'=>2,'the\_third'=>3);

echo $arr['the\_first'].'<br />';

echo $arr['the\_second'].'<br />';

echo $arr['the\_third'].'<br />';

?>

Створити масив з багатьма елементами можна також, оголосивши масив з одним елементом і додаючи до нього пізніше решту потрібних елементів:

<?php

$arr = array('the\_first'=>1);

$arr['the\_second']=2;

$arr['the\_third']=3;

foreach($arras $current)

echo $current.'<br />';

?>

Оскільки в асоціативних масивах індекси не є числами, то для роботи з такими масивами неможливо скористатися лічильником у циклі for. У цьому випадкові слід застосувати цикл foreach або конструкції list()та each().

<?php

$arr = array('the\_first'=>1,'the\_second'=>2,'the\_third'=>3);

foreach($arr as $current)

echo $current.'<br />';

?>

Якщо необхідно отримати доступ до індексів масиву, використовують конструкцію:

<?php

$arr = array('the\_first'=>1,'the\_second'=>2,'the\_third'=>3);

foreach($arr as $key => $value)

echo '$arr['.$key.']='.$value.'<br />';

?>

Результат:

$arr[the\_first]=1  
$arr[the\_second]=2  
$arr[the\_third]=3.

Наведений нижче код виводить вміст масиву за допомогою конструкції each():

<?php

$arr = array('the\_first'=>1,'the\_second'=>2,'the\_third'=>3);

while($element = each($arr))

{

echo '$arr['.$element['key'].']='.$element['value'].'<br />';

}

?>

У цьому прикладі змінна $element– це масив. При викликові функції each() вона повертає масив з чотирма значеннями і чотирма індексами. Комірки key та 0 містять ключ поточного елементу, value та 1 – його значення.

Важливо: при використанні функції each() слід пам’ятати, що масив відстежує поточний елемент. Якщо в одному і тому ж сценарії необхідно двічі скористатися значеннями одного і того ж масиву, то потрібно за допомогою функції reset() встановити поточний елемент на початок масиву.

Багатовимірні асоціативні масиви:

Багатовимірні асоціативні масиви можуть містити кілька ключів, відповідних конкретному індексу асоціативного масиву. Розглянемо приклад багатовимірного асоціативного масиву:

<?php

// Многомерныймассив

$A["Ivanov"]=array("name"=>"Іванов І.І.","age"=>"25","email"=>"ivanov@pu.if.ua");

$A["Petrov"]=array("name"=>"Петров П.П.","age"=>"34","email"=>"petrov@pu.if.ua");

$A["Sidorov"]=array("name"=>"Сидоров С.С.","age"=>"47","email"=>"sidorov@pu.if.ua");

?>

Доступ до елементів багатовимірного асоціативного масиву здійснюється таким чином:

echo $A["Ivanov"]["name"]; // Виводить Іванов І.І.  
echo $A["Petrov"]["email"]; // Виводить petrov@pu.if.ua

Асоціативні багатовимірні масиви можна створювати і класичним способом, хоча це не так зручно:

<?php

// Многомерныйассоциативныймассив

$A["Ivanov"]["name"]="Іванов І.І.";

$A["Ivanov"]["age"]="25";

$A["Ivanov"]["email"]="ivanov@pu.if.ua";

$A["Petrov"]["name"]="Петров П.П.";

$A["Petrov"]["age"]="34";

$A["Petrov"]["email"]="petrov@pu.if.ua";

$A["Sidorov"]["name"]="Сидоров С.С.";

$A["Sidorov"]["age"]="47";

$A["Sidorov"]["email"]="sidorov@pu.if.ua";

// Одержуємо доступ до багатовимірного асоціативного масиву

echo$A["Ivanov"]["name"]."<br>";// Виводить Іванов І.І.

echo$A["Sidorov"]["age"]."<br>";// Виводить 47

echo$A["Petrov"]["email"]."<br>";// Виводить petrov@pu.if.ua

?>

Видалити окремий елемент масиву можна за допомогою функції **unset(),** а перевірити існування масиву можна за допомогою функції **isset().** Визначимо масив з 10 елементів і знищимо кожен парний елемент.

<?php   
$arr = array(9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1 , 0);   
unset($arr[0], $arr[2], $arr[4], $arr[6], $arr[8]);   
// Перевіряєм чи існують елементи масиву   
for($i = 0 ; $i < 10 ; $i++)   
{   
if(isset($arr[$i])) echo "Елемент $arr[$i] визначений <br />" ;   
else echo "Елемент $arr[$i] не визначений <br />" ;   
}   
?>

Результатом роботи скрипта будуть наступні рядки

елемент $arr[0] не визначений   
елемент $arr[1] визначений   
елемент $arr[2] не визначений   
елемент $arr[3] визначений   
елемент $arr[4] не визначений   
елемент $arr[5] визначений   
елемент $arr[6] не визначений   
елемент $arr[7] визначений   
елемент $arr[8] не визначений   
елемент $arr[9] визначений

За допомогою функції **unset()** можна знищити весь масив відразу.

<?php   
$arr = array(9 , 8 , 7 , 6 , 5 , 4 , 3 , 2 , 1 , 0);   
unset($arr);   
if(isset($arr)) echo "Массив визначений" ;   
else echo "Массив не визначений" ;   
?>

До цього масиви виводилися за допомогою циклу, проте в PHP передбачена спеціальна функція для виведення вмісту масиву **print\_r().** Функція орієнтована на вивід в консольний потік, тому при виведенні результатів у вікно браузера краще обрамити її тегами <pre> і </ pre>:

<? Php   
$arr[] = "345";   
$arr[] = "mail@pu.if.ua";   
$arr[] = "http://www.softtime.ua";   
$arr[] = "login";   
$arr[] = "password";   
echo "<pre>";   
print\_r($arr);   
echo "</pre>";   
?>

Результат роботи скрипта виглядає наступним чином:

Array   
(  
[0] => 345   
[1] => mail@pu.if.ua   
[2] => http://www.softtime.ua   
[3] => login   
[4] => password   
)

### Операції над масивами

1) Визначення числа елементів в масиві **count():**

Створимо масив $name:

<?php   
$name = array('Boss' , 'Lentin' , 'NAV' , 'Endless' , 'Dragons' , 'SiLeNT' , 'Doctor' , 'Lynx');   
?>

Щоб визначити число елементів в масиві можна поступити наступним чином:

<?php   
echo 'Число елементов в массиве - ' . count($name);   
?>

результат:

Число елементов в массиве – 8

2) Об'єднання масивів

a) Створимо два асоціативних масиву $a і $b:

<?php   
$a = array(“a” => ”aa” , “b” => “bb”);   
$b = array(“c” => ”cc” , “d” => “dd”);   
?>

Нехай необхідно створити масив $c, які буде містити як елементи масиву $a так і масиву $b:

<?php   
$a = array("a" => "aa" , "x" => "xx");   
$b = array("c" => "cc" , "d" => "dd");   
$c = $a + $b ;   
echo "<pre>" ;   
print\_r($c);   
echo "</pre>" ;   
?>

результат:

Array   
(  
[a] => aa   
[x] => xx   
[c] => cc   
[d] => dd   
)

b) Створимо два числових масиву $a і $b:

<?php   
$a = array(10 , 20);   
$b = array(100 , 200 , 300 , 400 , 500);   
?>

Їх вже не вийде об'єднати за допомогою конструкції $c = $a + $b ;. Для їх об'єднання буде потрібно скористатися функцією **array\_merge():**

<?php   
$c = array\_merge($a , $b);   
?>

3) Сортування масиву

Скористаємося масивом $name:

<?php   
$name = array('Boss' , 'Lentin' , 'NAV' , 'Endless' , 'Dragons' , 'SiLeNT' , 'Doctor' , 'Lynx');   
?>

Нехай потрібно впорядкувати масив в алфавітному порядку, для цього можна скористатися наступним кодом:

<?php   
sort($name);   
echo "<pre>" ;   
print\_r($name);   
echo "</pre>" ;   
?>

результат:

Array   
(  
[0] => Boss   
[1] => Doctor   
[2] => Dragons   
[3] => Endless   
[4] => Lentin   
[5] => Lynx   
[6] => NAV   
[7] => SiLeNT   
)

Нехай необхідно з масиву $name вибрати найкоротший елемент(у якого найменша кількість символів), в цьому випадку можна скористатися кодом:

<?php   
$name = array('Boss' , 'Lentin' , 'NAV' , 'Endless' , 'Dragons' , 'SiLeNT' , 'Doctor' , 'Lynx');   
$min = strlen($name[0]);   
$nam = $name[0];   
for($i = 1 ; $i < count($name); $i++)   
{   
$len = strlen($name[$i]);   
if($len < $min)   
{   
$nam = $name[$i];   
$min = strlen($nam);   
}   
}   
echo 'Найменша довжина - ' . $nam ;   
?>

4) Переміщення всередині масиву

Створимо масив $num:

<?php   
$num = array(1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10);   
?>

Нехай потрібно відобразити елементи масиву в зворотному порядку, в цьому випадку можна скористатися кодом:

<?php   
$end = end($num);   
While($end)   
{   
echo $end . ' - ' ;   
$end = prev($num);   
}   
?>

результат:

10 - 9 - 8 - 7 - 6 - 5 - 4 - 3 - 2 – 1

Наведений вище код можна модифікувати:

<?php   
$num = range(1 , 10);   
print\_r(array\_reverse($num));   
?>

Функція **range(1,10)** створює масив(1,2,3,4,5,6,7,8,9,10) з випадковим розташуванням елементів. Функція **array\_reverse()** приймає масив і повертає елементи в зворотному порядку(10,9,8,7,6,5,4,3,2,1)

Функція **reset()** повертає покажчик на перший елемент в масиві. Скористаємося масивом $num:

<?php   
$num = array(1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10);   
?>

Нехай необхідно вивести всі елементи по порядку, і на останньому елементі масиву повернути покажчик на перший елемент масиву. Цю операцію можна здійснити за допомогою наступного коду:

<?php   
$i = 0 ; //Індекс 1 елемента   
while($i < count($num))   
{   
echo $num[$i]. ' ' ;   
$i++;   
//Перевірка якщо $i рівний числу елементів в масиві  
//тді виводим останній елемент і повертаєм вказівник   
if($num[$i] == count($num))   
{   
echo $num[$i];   
reset($num);   
echo '<br />' . "Кінець масиву" ;   
exit();   
}   
}   
?>

результат:

1 2 3 4 5 6 7 8 9 10   
Конець масиву

5) Перемішування елементів в масиві **shuffle()**

Функція **shuffle()** перемішує значення в масиві, і якщо масив асоціативний то повертає його як список:

<?php   
$a = array(43 , 'PHP' , 4 , 57 , 'Boss' , 90);   
shuffle($a);   
foreach($a as $n) echo "$n " ;   
?>

6) Випадковий елемент масиву

Якщо є готовий масив, з якого необхідно вивести один випадковий елемент, для цього необов'язково перемішувати весь масив за допомогою функції **shuffle(),** досить згенерувати випадковий індекс масиву:

<?php   
  
$arr = array(1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10);   
  
$index = rand(0 , count($arr) - 1);   
  
echo $arr[$index];   
?>

7) Отримання частини масиву **array\_slice()**

Створимо масив $a

<?php   
$a = array('a' , 'b' , '3' , '5' , 'f');   
?>

Отримати частину масиву можна за допомогою наступного коду:

<?php   
$b = array\_slice($a , 2) // вивід 3, 5, f   
$b = array\_slice($a , 0 , 3) // a, b, 3   
?>

8) Серіалізация масиву

Функції **serialize()** і **unserialize()** дозволяють здійснювати упакування і розпаковування, відповідно, масивів і об'єктів.

### зауваження

Серіалізация вперше з'явилася в об'єктно-орієнтованих бібліотеках,(першої з яких була MFC), потім серіалізація стала з'являтися в об'єктно-орієнтованих мовах(Java). Ідея серіалізації полягає в тому, що об'єкти і масиви дуже складні за своєю структурою і на збереження їх шляхом перебору кожного елемента потрібно значний обсяг коду - найпростішим рішенням є збереження таких структур у вигляді єдиної закодованої послідовності - байт-коді. У PHP функції серіалізації упаковують дані не у вигляді байт-коду, а у вигляді рядка.

<?php   
$poll[0] = 23 ;   
$poll[1] = 45 ;   
$poll[2] = 34 ;   
$poll[3] = 2 ;   
$poll[4] = 12 ;   
// Упаковуємо масив в рядок   
$str = serialize($poll);   
echo $str . "<br />" ;   
// Розпаковуємо масив із рядка   
$arr = unserialize($str);   
print\_r($arr);   
?>

результат:

a:5:{i:0;i:23;i:1;i:45;i:2;i:34;i:3;i:2;i:4;i:12;}

Array   
(  
[0] => 23   
[1] => 45   
[2] => 34   
[3] => 2   
[4] => 12   
)

Ви можете ознайомитися з усіма функціями для роботи з масивами у PHP в [довіднику функцій](http://www.php.su/functions/?cat=array).

**Передавання параметрів форми**

Спочатку напишемо HTML-документ, який буде містити практично всі елементи HTML-форми Параметри форми будемо передавати скрипту для наступної обробки. Отже, лістинг HTML-документа send.html:

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">

<title> Test </ title>

</ Head>

<body>

<h3> Тестова форма </ h3>

<form name="form1" method="post" action="script.php">

<p> <span> Текстове поле: </span>

<input type="text" name="textfield">

</p>

<p> Поле введення пароля:

<input type="password" name="pswfield">

</p>

<p> Приховане поле hidden

<input name="hidden" type="hidden" id="hidden" value="Приховане\_значення">

</p>

<hr size="1">

<p> Незалежні перемикачі(checkbox): </p>

<p>

<input type="checkbox" name="checkbox1" value="1">

Варіант перший

<input type="checkbox" name="checkbox2" value="1">

Варіант другий

<input type="checkbox" name="checkbox3" value="1" checked>

Варіант третій(за замовчуванням) </p>

<hr size="1">

<p> Залежні перемикачі(radio): </p>

<p>

<input name="radiobutton" type="radio" value="yes">

Так

<input name="radiobutton" type="radio" value="no">

Ні </p>

<hr size="1">

<p> Багаторядкове текстове поле(textarea): </p>

<p>

<textarea name="textarea" cols="40" rows="10"> Текст за умовчанням </ textarea>

</p>

<hr size="1">

<p> Список з єдиним вибором: </p>

<p>

<select name=day\_s size=1>

<option value=1> понеділок </option>

<option value=2> вівторок </option>

<option value=3 selected> середа </option>

<option value=4> четвер </option>

<option value=5> п'ятниця </option>

<option value=6> субота </option>

<option value=7> неділя </option>

</select>

</p>

<p> Список із множинним вибором(multiple): </p>

<p>

<select name=day\_m[] size=7 multiple>

<option value=1 selected> понеділок </option>

<option value=2> вівторок </option>

<option value=3> середа </option>

<option value=4> четвер </option>

<option value=5> п'ятниця </option>

<option value=6> субота </option>

<option value=7> неділя </option>

</select>

</p>

<hr size="1">

<p>

<input type="submit" value="Відіслати форму">

<input type="reset" value="Очистити форму">

</p>

</form>

</body>

</html>

Коли користувач натискає кнопку "Відіслати форму", браузер передасть скрипту наступні параметри:  
• textfield - значення текстового поля;  
• pswfield - значення поля введення пароля;  
• hidden - значення прихованого поля;  
• параметри checkbox: checkbox1, checkbox2 і checkbox3 будуть передані тільки в тому випадку, якщо відповідні їм незалежні перемикачі активні;  
• radiobutton - значення групи radio(буде передано одне із значень: Yes або No);  
• textarea - вміст багатострічкової текстової області;  
• day\_s - значення списку з єдиним вибором;  
• day\_m - значення списку із множинним вибором.  
  
Тепер перед нами стоїть завдання обробки всіх параметрів переданої форми за допомогою PHP скрипта.  
Параметри textfield, pswfield і textarea обробляються досить просто. Наприклад, для відображення значення параметра textfield достатньо написати в обробному скрипті: echo $\_POST['textfield'];  
З параметрами checkbox1, checkbox2, checkbox3, і radiobutton справа дещо складніша. Якщо перемикач не активний, то перелічені параметри взагалі не будуть передані на сервер, ніби їх взагалі не було.  
Отже, при спробі звернутися в скрипті до цих параметрів, ми отримаємо повідомлення, що змінна не існує. Тому просто написати echo $\_POST['checkbox1']; ми не можемо, нам необхідно спочатку перевірити існування цих параметрів в запиті.  
Перевірка існування параметра здійснюється за допомогою функції isset(), яка служить для перевірки існування змінних.  
if(isset($\_POST['checkbox1'])) echo $\_POST['checkbox1'];  
if(isset($\_POST['radiobutton'])) echo $\_POST['radiobutton'];  
Тільки після перевірки існування перерахованих параметрів форми можна починати роботу з змінними.  
  
  
Складніше обробляти параметри списку із множинним вибором, оскільки в цьому випадку параметри передаються так:  
day\_m = 01 & day\_m = 03 & day\_m = 07 ...  
Ми опинилися в ситуації, коли один параметр має кілька значень. Це нагадує нам масив даних. В цьому випадку множинний список можна представити у вигляді масиву, а обробити його елементи за допомогою циклу foreach. Навіть не обов'язково знати кількість елементів множини списку. Потрібно лише попередньо дати зрозуміти транслятору PHP, що ми будемо передавати масив:  
<select name="day\_m[]" size=7 mutiple>  
Квадратні дужки[] - це ознака масиву. Циклічна обробка масиву здійснюється так:  
foreach($\_POST['day\_m'] as $key => $value)  
echo "$key = $value ";  
А тепер наведемо остаточний лістинг PHP скрипта, обробника нашої тестової форми:

<? php

/ / Виводимо HTML-заголовки:

echo '<html>';

echo '<head>';

echo '<meta http-equiv="Content-Type" content="text/html; charset=windows-1251">';

echo '<title> Test </ title>';

echo '</ head>';

echo '<body>';

echo '<h3> Тестова форма </ h3>';

echo "<p> Передане значення текстового поля: <b>". $\_POST['textfield']. "</ b> </ p>";

echo "<p> Передане значення поля пароля: <b>". $\_POST['pswfield']. "</ b> </ p>";

echo "<p> Передане значення прихованого поля hidden: <b>". $\_POST['hidden']. "</ b> </ p>";

echo '<hr size="1">';

echo '<p> Були включені наступні незалежні перемикачі: </ p>';

if(isset($\_POST['checkbox1'])) echo "<p> <b> Перший </ b> </ p>";

if(isset($\_POST['checkbox2'])) echo "<p> <b> Другий </ b> </ p>";

if(isset($\_POST['checkbox3'])) echo "<p> <b> Третій </ b> </ p>";

echo '<hr size="1">';

if(isset($\_POST['radiobutton']))

{

echo '<p> Був обраний незалежний перемикач з наступним значенням:';

if($\_POST['radiobutton'] === "yes") echo "<b> Yes </b>";

if($\_POST['radiobutton'] === "no") echo "<b> No </b>";

echo '</p>';

}

else echo '<p> Жоден з незалежних перемикачів не був вибраний </p>';

echo '<hr size="1">';

echo '<p> Значення багатострічкового текстового поля: </p>';

echo "<p> <b>".$\_POST['textarea']. "</b> </p>";

echo '<hr size="1">';

echo "<p> Значення списку з єдиним вибором: <b>". $\_POST['day\_s']. "</b> </p>";

echo '<hr size="1">';

echo '<p> Значення списку із множинним вибором: </p>';

foreach($\_POST['day\_m'] as $keys => $values)

echo "<b> $values </b> <br>";

echo '<hr size="1">';

echo '</body>';

echo '</html>';

?>

**Завдання:**

1) Ознайомитися з описом масивів та функціями опрацювання масивів у PHP(див. теор. Відомості до 2-ї і 3-ї роботи).

2) Виконати всі пункти лабораторної роботи.

3) Надати звіт про виконану роботу.

**Хід роботи**

1. У масив внесіть 5 довільних чисел. Використовуючи конструкцію **foreach,** виведіть їх і їх квадрати у вигляді:

4^2=16

2^2=4

5^2=25 і т.д.

2.У HTML формі користувач вводить у шість різних полів: прізвище, ім'я, вік, e-mail і пароль і повторити пароль. Після натискання кнопки ***“Готово”*** запускається PHP-скрипт, який вносить ці дані в асоціативний масив і далі виводить їх в таблицю, використовуючи конструкцію **foreach**. Перед виводом передбачити перевірку, щоб кожне із переданих даних було не порожнім і паролі співпадали.

3. Створіть і заповніть багатовимірний асоціативний масив, який містить наступні дані по трьох країнах: назва, столиця, населення(дані придумайте самостійно).

Приклад реалізації одного із елементів багатовимірний асоційований масив:

$country["Ukraine"]=array("name"=>"Україна","capital"=>"Київ","popul"=>"45");

* використовуючи конструкцію **foreach** (для багатовимірного масиву одну конструкцію **foreach** вкладену в іншу)**,** виведіть з масиву:
* таблицю, яка містить три стовпці. Заголовки стовпців: «Назва», «Столиця», «Населення, млн.ч.». Кожен із стовпців містить відповідні дані по заданих країнах;К-сть рядків залежить від кількості заданих країн.
* три речення, типу «Столиця України — Київ, населення —45 млн. ч.;
* ключі першого та другого рівнів і їх значення, попередньо посортувавши елементи масиву $country:

Ukraine:

name=>Україна

capital=>Київ

popul=>45;

- виведіть даний масив у вікно бравзера, використовуючи функцію **print\_r().**

4. Створити анкету, яка містить чотири запитання, кожне з яких має різні види варіантів відповіді: радіокнопки, перемикачі, списки єдиного і множинного вибору. Створити скрипт, який перевіряє наявність відповіді на кожне із запитань і виводить текст запитання і вибрані до них відповіді, у випадку відсутності відповіді виводить в вікно браузера відповідне повідомлення.

**Контрольні запитання**

1. Наведіть приклад одного з можливих варіантів заповнення масиву значеннями у PHP.

2. Наведіть приклад одного з можливих варіантів заповнення асоційованого масиву значеннями у PHP.

3. Поясніть схему роботи оператора foreach у PHP.

4. Поясніть механізм передачі значень за посиланнями.

5. Наведіть приклади функцій, які можуть бути використані для опрацювання масивів у PHP.