Тема: «Функции для работы с массивами. Включение файлов»

Цель: Научиться создавать простые массивы и работать с ними. Ознакомиться со способом создания PHP шаблонов страницы

Теоретическая часть

Введение

Массивом называется переменная, имеющая упорядоченное множество значений. Порядок места расположения элементов в массиве определяется ключами (числа или символы). Значением элементов может быть любые тексты ил числовые значения. Для работы с массивами существует большое количество встроенных функций. Некоторые из них приведены в данной работе:

1) функция ***array()***; - создание массива.
2) функция ***list()***; - передача значений элементов массива в переменные.
3) цикл ***foreach*** - перебор массива по ключам.
4) функция ***array_merge()***; - объединение массивов.
5) функция ***array_slice ()***; - выделение части массива.
6) функция ***print_r ()***; - вывод массива на экран.
7) функции ***asort() и arsort()***; - сортировка элементов массива.
8) функция ***compact()***; - упаковывает переменные в массив.
9) функция ***unset()***; - удаляет элементы или сам массив.
10) функция ***count()***; - подсчитывает количество элементов в массиве.

Использование функций

Рассмотрим некоторые часто используемые функции для работы с массивами.

1) Функция array()

Функция Array() используется специально для создания массивов. При этом она позволяет создавать пустые массивы (Практическая работа №13).

2) Функция list()

Функция предназначена для передачи значений элементов массива в переменные
Исходный массив:

```
$names[0]="Александр";
$names[1]="Николай";
$names[2]="Яков";
```

Использование функции **list():**

```
list ($alex, $nick, $yakov) = $names;
```

Полученный результат:

```
$alex = "Александр";
$nick = "Николай";
$yakov = "Яков";
```

Если нам нужны только "Николай" и "Яков", то мы можем сделать так:

```
list (, $nick, $yakov) = $names;
```

3) Цикл перебора массивов foreach

Данный цикл предназначен специально для перебора массивов. Синтаксис цикла **foreach:**

Выводит массив с ключами	Выводит массив без ключей
foreach (массив as $ключ=>$значение) {команды};	foreach (массив as $значение) {команды};
<?php **$names["Иванов"] = "Андрей";** **$names["Петров"] = "Борис";** **$names["Волков"] = "Сергей";** **$names["Макаров"] = "Федор";** **foreach ($names as $key => $value) {** **echo "$value $key ";** **}** **?>**	**<?php** **$names[] = "Андрей";** **$names[] = "Борис";** **$names[] = "Сергей";** **$names[] = "Федор";** **foreach ($names as $value) {** **echo "$value ";** **}** **?>**
Андрей Иванов **Борис Петров** **Сергей Волков** **Федор Макаров**	**Андрей** **Борис** **Сергей** **Федор**

4) Функция array_merge();

Слияние (конкатенация) массивов - это операция создания массива, состоящего из элементов нескольких других массивов

```
$L1=array(100,200,300);
$L2=array(400,500,600);
$L=array_merge($L1,$L2);
```
// теперь $L===array(100,200,300,400,500,600);

5) Функция array_slice ();

Данная функция используется для получения части массива (разрез).
Первый параметр указывает на номер индекса, а второй на количество элементов:

```
$input = array ("a", "b", "c", "d", "e");
$output = array_slice ($input, 2); // "c", "d", "e"
$output = array_slice ($input, 2, -1); // "c", "d"
$output = array_slice ($input, -2, 1); // "d"
$output = array_slice ($input, 0, 3); // "a", "b", "c"
```
Отрицательные значения отсчитывают номер элемента от конца массива.

6) Функция print_r ();

Данная функция печатает читабельную информацию о переменной (массиве).
Пример:

```
<?php
$a = array('a' => 'apple',  'b' => 'banana',  'c' => array('x',
'y', 'z'));
echo "<pre>";
print_r($a);
```

```
echo "</pre>";
?>
```
выдаст на выводе:

```
Array
(
    [a] => apple
    [b] => banana
    [c] => Array
    (
        [0] => x
        [1] => y
        [2] => z
    )
)
<?php
```

7) Функции asort() и arsort();

Функция asort() сортирует массив, в алфавитном (если это строки) или в возрастающем (для чисел) порядке.
Пример:

```
$A=array("a"=>"Иван","b"=>"Сергей","c"=>"Александр","d"=>"Виктор")
;
asort($A);
foreach($A as $k=>$v) echo "$k=>$v ";
```
// выводит c=>Александр d=>Виктор a=>Иван b=>Сергей
// как видим, поменялся только порядок пар ключ=>значение

Функция arsort() выполняет то же самое, но она упорядочивает массив не по возрастанию, а по убыванию.

8) Функции compact();

Функция compact() упаковывает в массив переменные из текущего контекста (глобального или контекста функции).
Пример:

```
$a="Иванов";
$b="Алексеев";
$c="Викторов";
$A=compact("a","b","c");
echo "<pre>";
print_r ($A);
echo "</pre>";
```

выводит

```
Array
(
    [a] => Иванов
    [b] => Алексеев
    [c] => Викторов
```

```
)
```

9) Функция unset();

Данная функция удаляет элементы массива и сам массив.
Пример:

```
$arr = array ("a","b", "c", "d", "e", "f");
unset ($arr[3]);   //удалит элемент "d"

unset ($arr); //удалит массив $arr полностью
```

10) Функция count();

Данная функция предназначена для подсчета элементов массива.
Пример:

```
<?php
$arr[]=5;
$arr[]=4;
$arr[]=8;
$arr[]=3;
$arr[]=8;
echo "<h2>Число элементов массива: ".count($arr)."</h2>";
// Выводит: Число элементов массива: 5
?>
```

Включение файлов

Конструкции включений позволяют собирать PHP программу (скрипт) из нескольких отдельных файлов. В PHP существуют две основные конструкции включений: require и include. С их помощью можно создавать PHP шаблоны сайтов.

Конструкция включений require

Конструкция require позволяет включать файлы в сценарий PHP до исполнения сценария PHP.
Общий синтаксис require такой: **require имя_файла**;

Пример:
Файл **header.html**:

```
<html>
<head><title>It is a title</title></head>
<body bgcolor=green>
```

Файл **footer.html**:
© Home Company, 2005.

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

Файл **script.php** (включение файлов)

```
<?php
require "header.htm";
require "footer.htm";
?>
```

Результат:

```
<html>
```

```
<head><title>It is a title</title></head>
<body bgcolor=green>
</body>
</html>
```

Конструкция включений include

Конструкция **include** также предназначена для включения файлов в код сценария PHP. В отличие от конструкции **require** конструкция **include** позволяет включать файлы в код PHP скрипта во время выполнения сценария.

Общий синтаксис такой **include имя_файла**;

Конструкции однократного включения require_once и include_once

Используя конструкции однократного включения require_once и include_once, можно быть уверенным, что один файл не будет включен дважды.

Синтаксис данных функций такой: **require_once_имя файла** и **include_once_имя файла**.

Практическая часть

Задание 1

С помощью функции **array()** создайте ассоциативный массив со списком столиц государств (Россия – Москва, Франция – Париж, Германия – Берлин, Япония – Токио, Китай – Пекин, Австрия – Вена, Италия – Рим, Испания – Мадрид). С помощью функции **asort()** отсортируйте массив в алфавитном порядке по названию городов и выведите на экран.

Задание 2.

Используя оператор **foreach,** составьте список вашей группы и отсортируйте его по алфавиту. Выведите результат на экран.

Задание 3.

Создайте следующий массив: помидор, огурец, морковь, малина, смородина, клубника, апельсин, яблоко, груша. С помощью функции **array_slice ()** разделите этот массив на три массива: овощи, ягоды, фрукты. Выведите результат на экран.

Задание 4.

С помощью функции **array()** создайте массив из букв русского алфавита и подсчитайте их количество с помощью функции **count().**

Задание 5.

Создайте массив **$arr=array**('a', 'b', 'c', 'd', 'e'). С помощью одной переменной **$var** поменяйте местами элементы 'b' и 'c'.

Задание 6.

Создайте массив **$arr=array**('a', 'b', 'c', 'd', 'e'). С помощью одной переменной **$var** сделайте из него массив **array**('e', 'd', 'c', 'b', 'a').

Задание 7.

Даны два массива: первый с элементами '1', '2', '3', второй с элементами 'a', 'b', 'c'. Сделайте из них массив с элементами '1', '2', '3', 'a', 'b', 'c'.