Массивы

Массив — это набор значений, скрытых под одним именем. Получить доступ к конкретному значению можно по его номеру или текстовой строке.

Определение массива с помощью функции array()

```
$colors = array("red", "green", "blue", "gray");
```

Теперь обратимся к элементу массива по его номеру:

```
print "$colors[2]";
```

Создание элементов массива с помощью идентификатора

```
$colors[] = "red";
$colors[] = "green";
$colors[] = "blue";
$colors[] = "gray";
```

```
$colors[0] = "red";
$colors[200] = "green";
```

Создаем массив с помощью функции array() и добавляем к нему новый элемент:

```
$colors = array("red", "green", "blue", "gray");
$colors[] = "white";
```

Ассоциативные массивы

Ассоциативный массив — это массив, к элементу которого можно обратиться по имени. В других языках программирования аналогичные массивы называют структурами.

Создание ассоциативного массива

с помощью функции array()

```
$sal = array (
'name' => "Peel",
'city' => "London",
'comm' => 0.12
);
```

Теперь можно обратиться

к любому элементу массива:

```
print $sal['name'];
```

Непосредственное создание

ассоциативного массива

```
$sal['name'] = "Peel";
$sal['city'] = "London";
$sal['comm'] = 0.12;
```

Многомерные массивы

Многомерный массив — это массив, каждый элемент которого является массивом.

Для обращения ко второму элементу первого массива:

$my_array[1][2]

Апострофы у названий ключей использовать обязательно! ⟶

Создание многомерного массива

```
<html> <head>
<title> Создание многомерного массива </title>
</head> <body>
<?php  $sals = array (
   array ('name'=>"Peel",
         'city'=>"London",
         'comm'=>0.12
         ),
   array ('name'=>"Serres",
         'city'=>"San Jose",
         'comm'=>0.12
         ),
   array ('name'=>"Rifkin",
         'city'=>"Barcelona",
         'comm'=>0.15
        )
);
print $sals[0]['city']; // напечатается "London"
?>
</body> </html>
```

Работа с массивами

Получение размера массива

К любому элементу массива можно обратиться по его номеру:

print $colors[3];

Функция count() возвращает количество элементов массива:

$colors = array("red", "green", "blue", "gray");

print $colors[count($colors)-1];

Просмотр массива с помощью цикла

Оператор foreach:

```
foreach ($array as $ind)

    {   // тело цикла  }
```

```
<html> <head>
<title> Просмотр массива
</title> </head> <body>
<?php
$colors = array("red", "green", "blue", "gray");
foreach ($colors as $ind)
        {print "<p>$ind";
        }
?>
</body> </html>
```

Просмотр в цикле ассоциативного массива

Для просмотра ассоциативного массива в цикле:

```
foreach ($array as $key=>$value)
```

$array — имя массива,

$key — переменная, в которой сохраняется имя каждого элемента массива,

$value — переменная, где временно сохраняется значение каждого элемента.

Вывод многомерного массива

```
<html> <head>
<title> Просмотр многомерного массива в цикле
</title> </head>
<body>
<?php
$sals = array (  #1
          array ('name'=>"Peel",
                'city'=>"London",
                'comm'=>0.12
                ),
          array ('name'=>" Serres",
                'city'=>"San Jose",
                'comm'=>0.13
                ),
          array ('name'=>"Rifkin",
                'city'=>"Barcelona",
                'comm'=>0.15
                )
           );  #1
foreach ($sals as $ind)
   {  #2
    foreach ($ind as $key=>$val)
       {
        print "$key:  $val<br>";
       }
    print "<br>";
   }  #2
?>
</body> </html>
```

Результат:

name: Peel
city: London
comm: 0.12

name: Serres
city: San Jose
comm: 0.13

name: Rifkin
city: Barcelona
comm: 0.15

Управление массивами

Объединение массивов функцией array_merge()

```
<html> <head>
<title> Объединение массивов
</title> </head>
<body>
<?php
$first  = array("a", "b", "c");
$second = array("1", "2", "3");
$third = array_merge($first, $second);
foreach ($third as $val){
    print "$val<br>";
}
?>
</body> </html>
```

Добавление элементов к массиву с помощью функции array_push()

```
<html> <head>
<title> Добавление элементов к массиву
</title> </head>
<body>
<?php
$first = array("a", "b", "c");
$total = array_push($first, 1,2,3);
print "В массиве \$first всего $total элементов
<p>";
foreach ($first as $val){
   print "$val<br>";
}
?>
</body> </html>
```

Удаление первого элемента массива

```
<html> <head>
<title>Удаление первого элемента
      массива</title> </head> <body>
<?php
$an_array = array("a", "b", "c");
while (count($an_array)) {
   $val = array_shift($an_array);
   print "$val<br>";
   print "В массиве \$an_array есть"
      .count($an_array)." элементов<br>"; } ?>
```

```
a
В массиве $an_array есть 2 элементов
b
В массиве $an_array есть 1 элементов
c
В массиве $an_array есть 0 элементов
```

Выделение части массива с помощью функции array_slice()

Функции передается массив в первом аргументе, начальная позиция, или смещение от начала массива, и необязательный аргумент — длина участка. Если длина опущена, то считается, что нужно выделить часть массива от начальной позиции до его конца.

```
<html> <head>
<title> Выделение части массива
</title> </head> <body>
<?php
$first = array ("a", "b", "c", "d", "e", "f");
$second = array_slice($first, 2,3);
foreach ($second as $val){
  print "$val<br>";
}
?>
</body> </html>
```

Сортировка массивов

Сортировка простого массива с помощью функции sort()

```
<html> <head>
<title> Сортировка массива
</title> </head> <body>
<?php
$an_array = array("я", "а", "в", "е");
sort($an_array);
foreach ($an_array as $var) {
        print "$var<br>";
}
?>
</body> </html>
```

Простые массивы можно сортировать в обратном порядке. Для этого существует функция *rsort()*

Сортировка ассоциативного массива с помощью функции asort()

```
<html> <head>
<title> Сортировка ассоциативного массива
</title> </head> <body>
<?php
$first = array('first'=>5, 'second'=>2, 'third'=>1);
asort($first);
foreach ($first as $key=>$val){
    print "$key = $val<br>";
}
?>
</body> </html>
```

Для того чтобы отсортировать массив в обратном порядке, нужно воспользоваться функцией *arsort()*, которая работает точно так же.

Сортировка ассоциативного массива по именам полей с помощью функции ksort()

Функция *ksort()* принимает аргументом ассоциативный массив и сортирует его по именам полей. Она преобразует сортируемый массив и ничего не возвращает.

```
<html> <head>
<title> Сортировка ассоциативного массива по именам полей </title> </head> <body>
<?php
$first = array('x'=>5, 'a'=>2, 'f'=>1);
ksort($first);
foreach ($first as $key=>$val){
    print "$key = $val<br>";
}
?>
</body> </html>
```

Для сортировки массива в обратном порядке нужно воспользоваться функцией *krsort()*, которая работает точно так же.

Работа с формами

Программа для обработки ввода пользователя

```
<html> <head>
<title> Листинг 1. Простая HTML-форма </title>
</head> <body>
<!-- Форма для задания цвета через переменную
$bg. Ее использование — в скрипте 2.php -->

<p><b>Выберите цвет:</b></p>
<form action="2.php" method="GET">
<p><input type="radio" name="bg"
value="red">red</p>
<p><input type="radio" name="bg"
value="green">green</p>
<p><input type="radio" name="bg"
value="blue">blue</p>
<p><input type="submit"
value="Выполнить"></p>
</form>
</body> </html>
```

Мы создали форму, в которой есть набор кнопок-переключателей (radio button) с общим именем "bg" и кнопка передачи данных "submit". Атрибут action тега form указывает на файл 2.php, следовательно этот файл будет обрабатывать данные формы.

Обратите внимание, как передается значение переменной в методе GET (т.е. что пишется в поле «Адрес» браузера при нажатии кнопки «Выполнить») — это строка вида

2.php?bg=red

т.е. после имени выполняемого скрипта идет знак вопроса, затем имя передаваемой переменной, знак «равно» и значение этой переменной.

В листинге 2 приведена программа, обрабатывающая данные формы из листинга 1.

```
<html> <head>
   <title> Листинг 2. Чтение данных формы из листинга 1 </title>
   <style>
      body {
         background-color: silver;
         color: black;
      }
      a:link {
         color: white;
      }
      a:active {
         color: maroon;
      }
   </style>
   </head>
<body>
   <?php
   /* Из файла 1.php предается переменная $bg с названием цвета.
   Здесь она используется для задания цвета посещенной гиперссылки и цвета фона маленькой таблицы (прямоугольника)
   */
   $bg = $_GET["bg"];
   print "<style>a:visited {color: $bg}</style>";
   print "<p><table style='border: 1px; width: 100px; text-align: center'>\n";
   print "<tr> <td style='background-color: ${bg}'> <br> </td></tr></table>\n";
   print "<p style='text-align: center'><a href='ls10-1.php'>назад</a>";
   ?>
   </body>
</html>
```

Обработка элементов с многозначным выбором

```
<html> <head>
<title> Листинг 3. HTML-форма с выбором из списка
</title> </head> <body>
<!--
Здесь в скрипт 4.php передаются:
1) переменная $user
2) массив hobby[] со значениями, которые были выбраны
   в форме
-->
```

```
<form action="4.php" method="post">
<p>Введите ваше имя
<p><input type="text" name="user">
<p>Что вы любите делать в свободное время <br>
  (можно выбрать несколько вариантов)
<p><input type="checkbox" name="hobby[]"
   value="слушать музыку">слушать музыку
<p><input type="checkbox" name="hobby[]"
   value="читать книгу">читать книгу
<p><input type="checkbox" name="hobby[]"
   value="смотреть телевизор">смотреть телевизор
<p><input type="checkbox" name="hobby[]"
   value="гулять на улице">гулять на улице

<p><input type="submit" value="Выбор сделан">
</form>
</body> </html>
```

Листинг 4. Обработка данных формы из листинга 3

```
<html> <head>
<title> Листинг 4. Обработка данных формы
        из листинга 3 </title> </head> <body>
<?php
    $user = $_POST["user"];
    $hobby = $_POST["hobby"];
 print "<p>$user, оказывается, вы предпочитаете";
print "<ul>\n";
foreach ($hobby as $value){
     print "<li>$value\n";
}
print "</ul>\n";
?>
 </body> </html>
```

Несколько флажков с одним именем — не единственный способ, который позволяет ввести несколько значений. В листинге 3 можно заменить последовательность флажков с одним именем на тег SELECT с атрибутом multiple, и работать это будет точно так же.

```
<select name= "hobby[]" multiple> <option
value="лежать на диване">лежать на диване
<option value="читать книгу">читать книгу
<option value="смотреть телевизор">смотреть
телевизор
<option value="гулять на улице">гулять на улице
</select>
```

Доступ ко всем полям формы через ассоциативный массив

В ваше распоряжение предоставляется один из массивов — $_GET или $HTTP_POST.

```
<?php
/* Заменить цикл foreach в листинге 4.php
```

Листинг 5. Чтение данных произвольной формы с помощью ассоциативных массивов

```
*/
foreach ($_POST as $key=>$value) {
        print "$key = $value<br>\n";
}
?>
```

Если бы мы передали этой программе для обработки форму из листинга 3, то получили бы следующее:

```
user = guest
hobby = Array
```

Данные hobby существуют как элементы массива $ _POST [hobby], а мы попытались обратиться к ним как к простой переменной. В листинге 6 эта ошибка исправлена и, перед тем как вывести те или иные данные, их тип проверяется.

```
<?php
/* Заменить цикл foreach в листинге 4.php
```

Листинг 6. Чтение данных произвольной формы с проверкой типов

```
*/
foreach ($_POST as $key=>$value){
  if (gettype($value) == "array"){
     print "$key = <br>\n";
     foreach ($value as $v ){
          print "$v<br>";
     }
  }
  else{
     print "$key = $value<br>\n";
  }
}  ?>
```

Определение метода передачи

Листинг 7. Независимое от метода чтение данных

```
<?php
/* Заменить цикл foreach в листинге 4.php
   Листинг 7. Независимое от метода чтение данных
*/
$PARAMS = (isset($_POST))? $_POST : $_GET;
foreach ($PARAMS as $key=>$value){
  if (gettype($value) == "array"){
     print "$key = <br>\n";
     foreach ($value as $v)
     print "$v<br>";
  }
  else{
     print "$key = $value<br>\n";
  }
}
?>
```

Расположение HTML-текста и PHP-программы на одной странице

```
<html><head><title> Листинг 8. HTML-форма, вызывающая саму себя
   </title></head>
<body>
<?php

if (!isset ($_GET["bg"])) {
   $bg = "silver";
} else {
   $bg = $_GET["bg"];
}
print "<p>Выберите цвет фона прямоугольника";
print "<p><table style='border: 1px; width: 100px;'>";
print "<tr> <td style='background-color: $bg'> <br> ";
print "</td></tr></table>";
print "<form action='{$_SERVER['PHP_SELF']}' method='get'>";
?>
<p><input type="radio" name="bg" value="red">красный
<p><input type="radio" name="bg" value="green">зеленый
<p><input type="radio" name="bg" value="blue">синий
<p><input type="radio" name="bg" value="silver">исходный
<p><input type="submit" value="закрасить">
   </form></body>
</html>
```

Динамическое конструирование форм

```
<?php
if (isset ($_POST["bg"])) {
    $day = $_POST["day"];
    header("Location: $day");
    exit;
}
else { // начало блока else
?> <html> <head>
<title> Листинг 9. Посылка заголовка с помощью
        функции header() </title> </head> <body>
<?php
print "<form action='{$_SERVER['PHP_SELF']}' method='post'>";
?>
<select name="day">
<option value = "">Дни недели:
<option value='d1.htm'>Понедельник
<option value='d2.htm'>Вторник
<option value='d3.htm'>Среда
</select>
<input type="submit" value="Перейти">
</form>
<?php
    } // конец блока else
?>
</body> </html>
```

Послав браузеру заголовок "Location", вы перенаправите пользователя на новую страницу:
header ("Location: http://www.my_new_site.ru");

Сначала мы проверяем, было ли присвоено значение переменной $day, являющейся именем элемента select в форме. Если проверка дает положительный результат, вызывается функция header() с аргументом, в котором значение переменной $day присоединяется к строке "Location: ".

При передаче этой команды функция header() перенаправляет браузер на соответствующий файл (d1.htm, d2.htm или d3.htm). Если значение переменной $day, не задано, форма выводится в браузере.