Лекция 4. Массивы

Особенности массивов в PHP

Своеобразие PHP в том, что в нём массивы обладают следующими свойствами.

1. Все массивы - ассоциативные.
2. Если ключи - целые числа, то массивом можно пользоваться как массивом с числовыми индексами.
3. Элементы в массиве могут иметь разные типы.
4. Количество элементов в разных строках двумерного массива может быть разным.

Создание массива с числовыми индексами

Создадим массив из одного элемента

```
$m0=array('весна');
print "m0[0]=$m0[0]<BR>"; // m0[0]=весна
//Можно создать массив, присвоив значение его
элементу
$m1[0] = 3; //$m1 - новый массив
print "m1[0]=$m1[0]<BR>"; //m1[0]=3
```

Создадим массив из трёх элементов .

```
$M1 = array(2,7,12);
```

Добавим к нему ещё 2 элемента

```
$M1[3] = 'Маша';
$M1[4] = 2.71;
```

Добавим к нему сразу 7-й элемент, пропустив 5-й и 6-й, и распечатаем весь массив, считая, что в нём не 8, а 9 элементов:

```
$M1[7] = 100;
for($i=0; $i<=8; $i++)print "\$M1[$i] =
$M1[$i]<BR>";
/*напечатается
$M1[0] = 2
$M1[1] = 7
$M1[2] = 12
$M1[3] = Маша
$M1[4] = 2.71
$M1[5] =
$M1[6] =
$M1[7] = 100
$M1[8] =
В массиве 9 элементов, но 3 из них пустые.
*/
```

Ассоциативные массивы и оператор *foreach*

Создадим очевидный ассоциативный массив и распечатаем его.

```
$AssM['сезон'] = 'весна';
$AssM['месяц'] = 'май';
$AssM['число'] = 1;
foreach($AssM as $kluch => $znach) print "$kluch
- $znach<BR>;
/* Напечатается
   сезон - весна
   месяц - май
   число - 1
*/
```

Часто нужно перебрать все элементы массив, длина которого неизвестна, например, при чтении в массив текстового файла. Для этой цели служит оператор **foreach**. Существую две разновидности foreach. Приведённый выше оператор служит для перебора массива, рассматриваемого, как ассоциативный. Распечатаем массив $AssM, как массив с числовыми индексами, используя другую разновидность foreach.

```
foreach($AssM as $znach) print "$znach<BR>";
/* Напечатается
   весна
   май
   1
*/
```

Более нагляден следующий способ создания ассоциативного массива:

```
$cveta=array(
             "red"=>"Красный",
             "yellow"=>"Жёлтый",
             "green"=>"Зеленый",
             "blue"=>"Синий",
             "maroon"=>"Каштановый"
            );
```

Распечатаем как ассоциативный массив $M1, который создавался как массив с числовыми индексами .

```
foreach($M1 as $kluch => $znach) print "$kluch -
$znach<BR>;
/* Напечатается
   0 - 2
   1 - 7
   2 - 12
   3 - Маша
   4 - 2.71
   7 - 100
*/
```

Операции с массивами

- + - объединение по ключам;
- == - true, если $a и $b содержат одинаковые элементы;
- === - true, если $a и $b содержат одинаковые элементы, расположенные в одинаковом порядке;
- != - true, если в $a и $b есть неодинаковые элементы;
- <> - то же, что и !=
- !== - true, если $a и $b содержат одинаковые элементы, но порядок элементов разный;

Примеры

Объединение по ключам.

```
$a = array(1,2);
$b = array(3,4);
//У этих массивов ключи имеют одинаковые значения 0 и 1
$c = $a + $b;
foreach($c as $x)print "$x ";
// Результат 1 2
// Элементы с повторяющимися ключами берутся только один раз
//
$d = array(2 => 'a', 3 => 'b', 8 => 'c')
// Ключи в $a - 0, 1; ключи в $d - 2, 3, 8. Совпадающих ключей нет
$c = $a + $d; //$c=(1,2,'a','b','c')
```

Сравнение массивов

```
$b = $a; //Копирование массива. $b и $a одинаковые массивы
$g = $a == $b; //$g = 1, т.е. "истина"
$g = $a === $b; //$g = 1, т.е. "истина"
$c = array(0 => 2, 1 => 1);
$g = $a ==  $c; //$g = 1, т.е. "истина"
$g = $a ===  $c; //$g = 0, т.е. "ложь", не идентично
```

Функции, для работы с массивами

В PHP огромное количество встроенных функций. В руководстве по PHP, выпущенном в 2005 году, описывается 5211 функций. Только на просмотр такого количества нужно 5211/4/8 = 163 рабочих дня, если на каждую функцию тратить по 15 минут. И это многообразие всё время пополняется. Функций для работы с массивами - 80. Рассмотрим лишь некоторые из них.

count(имя_массива) - функция, возвращающая количество элементов в массиве. Подсчитаем длину рассмотренного выше массива $M1:

```
$L = count($M1); //$L=6
```

Из примера следует, что функция count возвращает количество непустых элементов массива.

Функции сортировки

- sort(имя_массива) - сортировка по возрастанию элементов массива с числовыми индексами.
- asort(имя_массива) - сортировка ассоциативного массива по возрастанию значений.
- ksort(имя_массива) - сортировка ассоциативного массива по возрастанию ключей.
- rsort(имя_массива) - сортировка по убыванию элементов массива с числовыми индексами.
- arsort(имя_массива) - сортировка ассоциативного массива по убыванию значений.
- krsort(имя_массива) - сортировка ассоциативного массива по убыванию ключей.
- usort(имя_массива) - определяемая пользователем функция сортировки массива с числовыми индексами.
- uasort(имя_массива) - определяемая пользователем функция сортировки ассоциативного массива по значениям.
- uksort(имя_массива) - определяемая пользователем функция сортировки ассоциативного массива по ключам.

Примеры

Сортировка по возрастанию элементов массива с числовыми индексами.

```
sort($M1);
for($i=0; $i<9; $i++)print "\$M1[$i] =
$M1[$i]<BR>";
/*Результат сортировки
$M1[0] = Маша
$M1[1] = 2
$M1[2] = 2.71
$M1[3] = 7
$M1[4] = 12
$M1[5] = 100
$M1[6] =
$M1[7] =
$M1[8] =
*/
```

Сортировка ассоциативного массива $cveta по возрастанию значений

```
asort($cveta);
foreach($cveta as $kluch=>$znach)
    print "$kluch - $znach<BR>";
/*Результат сортировки
```

```
  yellow - Жёлтый
  green -  Зеленый
  maroon - Каштановый
  red -    Красный
  blue -   Синий
*/
```

Сортировка ассоциативного массива $cveta по возрастанию ключей

```
ksort($cveta);
foreach($cveta as $kluch=>$znach)
     print "$kluch - $znach<BR>";
/*Результат сортировки
   blue -   Синий
   green -  Зеленый
   maroon - Каштановый
   red -    Красный
   yellow - Жёлтый
*/
```

Функции для добавления и удаления элементов массива

- array_pad - увеличить размер массива до заданной величины.
- array_pop - извлечь последний элемент массива.
- array_push - добавить один или несколько элементов в конец массива.
- array_shift - извлечь первый элемент массива.
- array_unshift - добавить один или несколько элементов в начало массива.

Часто применяется функция *array_shift(array)*, которая извлекает значение первого элемента массива *array* и возвращает это значение, сокращая размер *array* на один элемент. Все числовые ключи будут изменены таким образом, что нумерация массива начнётся с нуля, в то время как строковые ключи останутся прежними. Если array пуст (или не является массивом), будет возвращён *NULL*.

Эту функцию удобно использовать для удаления из массива и запоминания в отдельной переменной шапки таблицы.

Пример

```
  $Arr = Array('a'=> 'alfa','b'=> 'beta','c'=>
'gamma');
  $a1 = array_shift($Arr);
  print_r ($Arr); //print_r - функция для отладочной
печати
  print "\$a1=$a1<BR>";
//Результат
// Array ( [b] => beta [c] => gamma )
//  $a1=alfa
```

Двумерные массивы

Создадим двумерный массив

```
$M2= array(
          array(1,2,3),
          array(4,5,6,'семь'),
          array('a','b','c'),
         );
```

Можно создавать двумерный массив поэлементно.

```
$M3[0][0]='alfa';
$M3[0][1]='beta';
```

Подсчитаем количество строк в массиве $M2 и количество элементов в строке M2[1].

```
$Lrow = count($M2); // $Lrow = 3
$Lcol = count($M2[1]); // $Lcol = 4
```

Распечатаем двумерный массив $M2.

```
foreach($M2 as $row)
{  foreach ($row as $el)
   { print "$el ";
   }
   print '<BR>';
}
/* Результат
   1 2 3
   4 5 6 семь
   a b c
*/
```

Создадим явный ассоциативный массив, добавив в $M3 строку

```
$M3['odin'][0] = 'два';
$M3['odin'][1] = 'три';
print 'L='.count($M3)." \$M3['odin'][1] =
".$M3['odin'][1]."<BR>";
/* Результат
   L=2 $M3['odin'][1] = три
*/
```

Пример сортировки двумерного массива с помощью функции usort(). Сортируется массив $M2 по второму столбцу (с индексом 1) в порядке возрастания

```
usort($M2, 'compare1');
foreach($M2 as $row)
{  foreach ($row as $el)
   { print "$el ";
   }
   print '<BR>';
}
```

```
/* Результат
  a b c
  1 2 3
  4 5 6 семь
*/
 function compare1($x,$y) //Сортировка по возрастанию
 {                        //$x и $y - две соседние
строки
    if($x[1] == $y[1]) return 0; //порядок не
определён
    elseif(($x[1] > $y[1]) return 0; //первым $y[1]
    else                   return 1; //первым $x[1]
  }
```

Массив ассоциативный или с числовыми индексами?

Если где-нибудь что-нибудь убавится,
то где-то что-то прибавится.
Закон сохранения

В PHP все массивы ассоциативные, то есть, форально в PHP нет, в отличие от большинства языков высокого уровня, массивов с числовыми индексами. Согласно приведённому в шутливой фоме в эпиграфе данного параграфа закону сохранения, такое решение должно привести и к положительным и к отрицательным последствиям.

Достоинства ассоциативных массивов. Применение ассоциативных массивов вместо массивов с числовыми индексами в некоторых случаях позволяет сильно упростить программу, например программу обработки сообщений в узле связи. Следует отметить, что такие случаи на практике встречаются нечасто.

Недостатки ассоциативных массивов по сравнению с массивами с числовыми индексами:

- занимают больше места в памяти;
- большее время выборки элемента;
- очень сложные, скрытые от прикладной программы, алгоритмы записи и выборки.

Рассмотрим пример, в котором будем испльзовать массив *$MC* как массив с числовыми индексами. Создадим два первых элемента:

```
$MC[0] = 'синий';
$MC[1] = 'зелёный';
```

Подсчитаем количество элементов в массиве и распечатаем его с помощью оператора *for*, используемого только в случае, когда массив рассматривается как масссив с числовыми индекксами.

```
$L=count($MC);
for ($i=0; $i < $L; $i++) print "$i $MC[$i]<BR>";
```

```
/* Результат
0  синий
1  зелёный
*/
```

Предположим, что программа, в которой используется массив *$MC*, большая, разрабатывается долго и программист в какой-то момент забыл, что использует *$MC* как масссив с числовыми индексами, и добавил новый элемент с текстовым ключом:

```
$MC['red'] = 'Красный';
```

Снова подсчитаем количество элементов и распечатаем массив *$MC*

```
$L=count($MC);
for ($i=0; $i < $L; $i++) print "$i $MC[$i]<BR>";
/* Результат
0  синий
1  зелёный
2
*/
```

Так как мы условились, что в рассматриваемой программе *$MC* используется только как масссив с числовыми индексами, то элемент *$MC['red']* со значением 'Красный' никогда не будет использоваться. Его, как бы, нет. Такая ошибка очень трудно обнаруживается, но возникла она вследствии упрощения языка. В одном месте убавилось, в другом прибавилось, причём очень неприятно прибавилось.

Пустые и полупустые элементы массива

Элементы используемых в большинстве языков программирования настоящих массивов с числовыми индексами располагаются в последовательных ячейках памяти. Если массив, состоящий из *m* элементов, расположен, начиная с ячейки с адресом *n*, то адрес *i-го* элемента равен *n* + *i*. Таким образом, для того чтобы найти *i-й* элемент массива, нужно хранить только адрес начала и длину массива с числовыми индексами. Индексы элементов такого массива , в отличие от ключей ассоциативного массива, хранить в памяти ЭВМ не нужно.

Любая ячейка памяти в любой момент времени содержит какой-то набор нулей и единиц, то есть, физически пустых ячеек быть не может. Семантически (логически) элементы массива могут быть пустыми, например. в *i-й* элемент массива цен не занесена цена товара. Для указания, что элемент массива пуст, служит специальная величина *NULL*, хранимая в этом элементе.

Раасмотрим, что понимается под "пустым" элементом массива в PHP. Создадим массив $M:

```
$M[0]=100;
$M[1]=200;
$M['red']='красный';
$M[3]=NULL;
```

```
$M[4]=FAlSE;
$M[5]='';
$M[8]=300;
```

Распечатаем этот массив тремя способами.
1. Как массив с числовыми индексами.

```
for ($i=0; $i < 9; $i++) print "$i $M[$i]<BR>";
```

```
Результат
0 100
1 200
2
3
4
5
6
7
8 300
```

Элемент с ключом 'red' не виден, элементы со значениями *NULL, FALSE, '' (две кавычки)* и элементы *$M[2], $M[6], $M[7]* - пустые

2. С помощью оператора *foreach* распечатаем только значения элементов.

```
$i=0;
foreach(($M) as $x ) print $i++." $x<BR>";
```

```
Результат
0 100
1 200
2 красный
3
4
5
6 300
```

3. С помощью оператора *foreach* распечатаем все элементы массива и ключи.

```
foreach(($M) as $k => $z ) print "$k $z<BR>";
```

```
Результат
Ключ  Значение
 0       100
 1       200
 red     красный
 3
 4
 5
 8       300
```

Из сравненния действительного содержимого массива *$M* с результатами вывода на печать следует, что элементы с индексами 2, 6, 7 не существуют, а элементы 3, 4, 5 имеют ключи, но не имеют значений.