**Лекция 13.**

**Тема: 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](https://i.voenmeh.ru/kafi5/Kam.loc/PHPzadan/LEKCII_PHP.html#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.

В этом коротком обзоре перечислены далеко не все проблемы, возникающие из-за отсутствия в *PHP* явного типа массивов с числовыми индексами.