**Лекция 12.**

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

Cортировка 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] =

\*/

Cортировка ассоциативного массива [$cveta](https://i.voenmeh.ru/kafi5/Kam.loc/PHPzadan/LEKCII_PHP.html#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

\*/

Cортировка 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