Управляющие структуры

Условные операторы

В языке PHP два условных оператора: **if** и **?**.
Существует три типа оператора **if**. Первый тип - базовый условный оператор. Например:

```
if ($a > $b){
  echo "A больше Б";
}
```

Альтернативный синтаксис оператора **if**:

```
if ($a > $b):
  echo "A больше Б";
endif;
```

Второй тип - условный оператор **if-else**. Например:

```
if ($a > $b){
  echo "A больше Б";
}else{
  echo "A не больше Б";
}
```

Альтернативный синтаксис оператора **if-else**:

```
if ($a > $b):
  echo "A больше Б";
else:
  echo "A не больше Б";
endif;
```

Третий тип - условный оператор **if-elseif**. Например:

```
if ($a > $b){
  echo "A больше Б";
}elseif ($a == $b)
{
  echo "A равно Б";
}else{
  echo "A меньше Б";
}
```

Альтернативный синтаксис оператора **if-elseif**:

```
if ($a > $b):
  echo "А больше Б";
elseif ($a == $b):
  echo "А равно Б";
else:
  echo "А меньше Б";
endif;
```

Условный оператор **?** возвращает одно из двух значений, разделенных двоеточием. Использование оператора **?** может сделать более компактным текст программы. Например:

```
$text = ($a == $b) ? "А равно Б" : "А не равно Б";
echo $text;
```

Оператор выбора

Оператор выбора **switch** оценивает одно выражение и в зависимости от его значения выполняет один из нескольких блоков программы. Выражение в операторе **switch** чаще всего бывает простой переменной. Например:

```
switch ( $a ){
  case 1:
     echo "А равно 1";
     break;
  case 2:
     echo "А равно 2";
     break;
  case 3:
     echo "А равно 3";
     break;
  default:
     echo "А не равно ни 1, ни 2, ни 3";
}
```

Альтернативный синтаксис оператора **switch**:

```
switch ( $a ):
  case 1:
      echo "A равно 1";
      break;
  case 2:
      echo "A равно 2";
      break;
  case 3:
      echo "A равно 3";
      break;
  default:
      echo "A не равно ни 1, ни 2, ни 3";
endswitch;
```

Цикл с параметром

Цикл с параметром **for** относится к наиболее старому и заслуженному виду цикла.

```
<html>
<head>
 <title>Цикл for</title>
</head>
<body>
<?php
for ($a = 11; $a <= 19; $a++){
   echo "квадрат $a равен ".($a*$a)."<br>";
}
?>
</body>
</html>
```

Альтернативный синтаксис оператора **for**:

```
for ($a = 11; $a <= 19; $a++):
   echo "квадрат $a равен ".($a*$a)."<br>";
endfor;
```

Циклы с условием

В языке PHP существует два типа цикла с условием:

- **while** - цикл с предусловием;
- **do .. while** - цикл с постусловием.

Оператор **while** оценивает значение условия и, если оно истинно, выполняет действия в фигурных скобках (тело цикла). Как только значение условия станет ложным, выполнение цикла прекращается.

```
<html>
<head>
 <title>Цикл while</title>
</head>
<body>
<?php
$a = 11;
while ( $a <= 19 ){
   echo "квадрат $a равен ".($a*$a)."<br>";
   $a++;
}
?>
</body>
</html>
```

Альтернативный синтаксис оператора **while**:

```
while ( $a <= 19 ):
   echo "квадрат $a равен ".($a*$a)."<br>";
   $a++;
endwhile;
```

Цикл **do .. while** отличается от цикла **while** лишь тем, что истинность условия проверяется не *до*, а *после* выполнения тела цикла.

```
<html>
<head>
 <title>Цикл do .. while</title>
</head>
<body>
<?php
$a = 11;
do {
   echo "квадрат $a равен ".($a*$a)."<br>";
   $a++;
}while ( $a <= 19 );
?>
</body>
</html>
```

В организации цикла могут участвовать еще два оператора: **break** (выход из цикла) и **continue** (переход на следующий шаг).

Использование массивов

Основное назначение массивов в PHP - организация групп связанных значений. Каждый элемент массива имеет индекс (ключ) и значение. Индекс элемента массива указывается в квадратных скобках после имени массива. Для того, чтобы обратиться к пятому элементу массива $array1, надо написать: $array1[5]

Помните, что по умолчанию массив начинается не с первого элемента, а с нулевого.

Индекс может быть как числом, так и текстовой строкой. Массив со строковыми индексами называют *ассоциативным*, а сами индексы - именами элементов. Например, цены на товары хранятся в ассоциативном массиве $prices, индексами которого являются наименования товаров. Чтобы получить значение цены на конкретный товар, надо написать: $prices["яйца 10 шт. 1 сорт"]

Как вы заметил то ключом выступает название товара на русском с пробелами. Желательно такого не делать. Но такая запись вполне допустима.

Значение элемента массива может иметь любой тип. Возможна организация многомерных массивов, так как элемент массива может в свою очередь являться массивом. Примеры обращений к элементам многомерных массивов:

$array3[0][4][1]
$цены["овощи"]["помидор"]
$список[1]["фамилия"]

Создание массива

Массив можно создать с помощью функции **array()**, параметры которой и составляют массив. Параметры могут задаваться парами "ключ=>значение". Если при создании массива ключ не указывается, то индекс определяется положением элемента в массиве (начиная с 0). Например:

```
$рост = array (174, 181, 166); //Массив с индексацией, начинающейся с нуля
$цена = array ("помидоры" => 15, "огурцы" => 12); //Ассоциативный массив
$данные = array (
          "Иванов" => array ("рост" => 174, "вес"  => 68),
          "Петров" => array ("рост" => 181, "вес"  => 90),
          "Сидоров" => array ("рост" => 166, "вес"  => 73)
); //Двухмерный массив
```

Массивы можно создать и другим способом - непосредственно. Например:

```
$фрукты[] = "яблоко";
$фрукты[] = "груша";
$фрукты[] = "слива";
$цена["помидор"] = 15;
$цена["огурец"] = 12;
```

Индексы элементов неассоциативного массива можно не указывать. PHP автоматически вычислит их. Если же указать индексы таким образом:

```
$фрукты[0] = "груша";
$фрукты[5] = "яблоко";
```

то в массиве будет два элемента, причем последний с индексом 5. Элементы 1 - 4 не инициализируются.
Можно создать массив с помощью функции array(), а затем добавить к нему новый элемент:

```
$фрукты = array("яблоко","груша","слива");
...
$фрукты[] = "персик";
```

Подсчет количества элементов

Количество элементов в массиве можно определить с помощью функций **count()** или **sizeof()**.

ПРИМЕР:

```
<html>
<head>
<title>Размер массива</title>
</head>
<body>
<?php
$фрукты = array("яблоко", "груша", "слива", "персик");
echo "Размер массива \$фрукты равен ".count($фрукты)."<br>";
echo "Последний элемент массива \$фрукты - ".$фрукты[count($фрукты)-1]."<br>";
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

Размер массива $фрукты равен 4
Последний элемент массива $фрукты - персик

Для доступа к последнему элементу надо вычесть 1 из размера массива, так как индексация массива начинается с нуля. Для вывода зарезервированного символа "$" перед знаком доллара стоит символ обратной косой черты "\".
Частоту вхождения элементов в массив можно определить с помощью функции **array_count_values()**. Эта функция возвращает массив, в котором ключами являются элементы исследуемого массива, а значениями - частоты их вхождения в исследуемый массив.

ПРИМЕР:

```
<html>
<head>
<title>Размер массива</title>
</head>
<body>
<?php
$фрукты = array("яблоко", "груша", "слива", "персик", "груша");
print_r (array_count_values($фрукты));
?>
</body>
</html>
```

РЕЗУЛЬТАТ:
```
Array ( [яблоко] => 1 [груша] => 2 [слива] => 1 [персик] => 1 )
```

Функция **print_r()** отображает ключи и значения массива, указанного в аргументе.

Просмотр массива в цикле

Для итерационного просмотра содержимого массива служит функция **foreach**. С ее помощью можно просмотреть и простой (проиндексированный числами) массив, и ассоциативный, и многомерный.

ПРИМЕР:

```
<html>
<head>
<title>Просмотр массива</title>
</head>
```

```
<body>
<?php
$фрукты = array("яблоко", "груша", "слива", "персик", "груша");
foreach ($фрукты as $фрукт){
  echo  "$фрукт<br>";
}
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

яблоко
груша
слива
персик
груша

ПРИМЕР:

```
<html>
<head>
<title>Просмотр ассоциативного массива</title>
</head>
<body>
<?php
$цена = array ("помидоры" => 15, "огурцы" => 12);
foreach ($цена as $овощи => $руб)
  {
  echo  "$овощи стоят $руб руб.<br>";
  }
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

помидоры стоят 15 руб.
огурцы стоят 12 руб.

print_r - Вывод всех эллементов массива. Удобная функция для отладки.

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

РЕЗУЛЬТАТ:

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

Управление массивами

Для управления массивами в PHP существует целый ряд специализированных функций. Наиболее употребимыми являются:
array_merge() - объединение (слияние) массивов. Последующее значение элемента перекрывает предыдущее, если ключи строковые и одинаковые.

```
<html>
<head>
<title>Слияние массивов</title>
</head>
<body>
<?php
$мнение1 = array (1, "Катя" => "умная", 2, "Женя" => "красивая", 3);
$мнение2 = array (4, "Женя" => "глупая", 5, "Катя" => "милая");
echo  "1 + 2:<br>";
print_r (array_merge ($мнение1, $мнение2));
echo  "<br>2 + 1:<br>";
print_r (array_merge ($мнение2, $мнение1));
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

```
1 + 2: Array ( [0] => 1 [Катя] => милая [1] => 2 [Женя] => глупая [2] => 3 [3] => 4 [4] => 5 )
2 + 1: Array ( [0] => 4 [Женя] => красивая [1] => 5 [Катя] => умная [2] => 1 [3] => 2 [4] => 3 )
```

array_shift() - удаление первого элемента.

```
<html>
<head>
<title>Удаление первого элемента массива</title>
</head>
<body>
<?php
$фрукты = array("яблоко", "груша", "слива", "персик");
echo  "В вазе лежали:";
foreach ($фрукты as $tmp){
  echo  " $tmp";
}
echo  "<br>Первым съели ".array_shift($фрукты)."<br>";
echo  "В вазе остались:";
foreach ($фрукты as $tmp){
  echo  " $tmp";
}
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

В вазе лежали: яблоко груша слива персик
Первым съели яблоко
В вазе остались: груша слива персик

array_pop() - удаление последнего элемента.

```
<html>
<head>
<title>Удаление последнего элемента массива</title>
</head>
<body>
<?php
$фрукты = array("яблоко", "груша", "слива", "персик");
echo  "В вазе лежали:";
foreach ($фрукты as $tmp){
  echo  " $tmp";
}
echo  "<br>Съели ".array_pop($фрукты)."<br>";
echo  "В вазе остались:";
foreach ($фрукты as $tmp){
  echo  " $tmp";
}
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

В вазе лежали: яблоко груша слива персик
Съели персик
В вазе остались: яблоко груша слива

array_push() - добавление элементов в конец массива.

```
<html>
<head>
<title>Добавление элементов в конец массива</title>
</head>
<body>
<?php
$a = array(1, 2, 3);
echo  "Массив \$a:";
foreach ($a as $tmp) echo " $tmp";
$k = array_push($a, 4, 5);
echo  "<br>Массив \$a после добавления:";
foreach ($a as $tmp) echo " $tmp";
echo  "<br>Его длина = $k";
?>
</body>
</html>
```

РЕЗУЛЬТАТ:
Массив $a: 1 2 3
Массив $a после добавления: 1 2 3 4 5
Его длина = 5

Имейте в виду, что если в качестве второго аргумента функции array_push() передать массив, то этот массив будет добавлен как элемент, т.е. будет создан двумерный массив.

array_unshift() - добавление элементов в начало массива.

```
<html>
<head>
<title>Добавление элементов в начало массива</title>
</head>
<body>
<?php
$a = array(1, 2, 3);
echo  "Массив \$a:";
foreach ($a as $tmp) echo " $tmp";
$k = array_unshift($a, 4, 5);
echo  "<br>Массив \$a после добавления:";
foreach ($a as $tmp) echo " $tmp";
```

```
echo  "<br>Его длина = $k";
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

```
Массив $a: 1 2 3
Массив $a после добавления: 4 5 1 2 3
Его длина = 5
```

array_slice() - выделение фрагмента массива.
Имеет три параметра: сам массив, смещение и длину выделяемого фрагмента. При положительном смещении отсчет выполняется от начала массива, при отрицательном - от конца. При положительной длине результирующий фрагмент будет содержать заданное число элементов. При отрицательной длине последним элементом фрагмента станет тот, который находится на указанном расстоянии от конца массива. Если параметр опущен, то фрагмент будет содержать все элементы от начального смещения и до конца массива.

```
<html>
<head>
<title>Выделение фрагмента массива</title>
</head>
<body>
<?php
$a = array("a", "b", "c", "d", "e", "f");
echo  "\$a:"; foreach ($a as $t) echo " $t";
$b = array_slice($a, 2);
echo  "<br>array_slice(\$a, 2):"; foreach ($b as $t) echo " $t";
$b = array_slice($a, 2, -1);
echo  "<br>array_slice(\$a, 2, -1):"; foreach ($b as $t) echo " $t";
$b = array_slice($a, -2, 1);
echo  "<br>array_slice(\$a, -2, 1):"; foreach ($b as $t) echo " $t";
$b = array_slice($a, 0, 3);
echo  "<br>array_slice(\$a, 0, 3):"; foreach ($b as $t) echo " $t";
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

```
$a: a b c d e f
array_slice($a, 2): c d e f
array_slice($a, 2, -1): c d e
array_slice($a, -2, 1): e
array_slice($a, 0, 3): a b c
```

Сортировки

Сортировать можно как простые, так и ассоциативные массивы. Для сортировки массивов в PHP существуют определенные функции:

- **sort()** - сортирует массив в алфавитном порядке, если хотя бы один из его элементов является строкой, и в числовом порядке, если все его элементы - числа.
- **rsort()** - работает как **sort( )**, но в обратном порядке.
- **asort()** - сортирует ассоциативный массив; работает как **sort( )**, но сохраняет имена элементов.
- **arsort()** - работает как **asort( )**, но в обратном порядке.
- **ksort()** - сортирует ассоциативный массив по именам элементов.
- **krsort()** - работает как **ksort( )**, но в обратном порядке.

```
<html>
<head>
<title>Сортировка ассоциативного массива</title>
</head>
<body>
<?php
$a = array("первый" => 6, "второй" => 2, "третий" => 1);
echo  "\$a:<br>"
asort ($a);
echo  "asort (\$a):<br>"
foreach ($a as $k => $t) echo " $k = $t<br>";
ksort ($a);
echo  "ksort (\$a):<br>"
foreach ($a as $k => $t) echo "$k = $t<br>";
?>
</body>
</html>
```

```
РЕЗУЛЬТАТ:
$a:
первый = 6
второй = 2
третий = 1

asort ($a):
третий = 1
второй = 2
первый = 6


ksort ($a):
```

второй = 2
первый = 6
третий = 1

Поиск элемента

Для проверки наличия элемента в массиве существуют функции:

- **in_array()** - если элемент найден, возвращает true, иначе - false.
- **array_search()** - если элемент найден, возвращает его ключ, иначе - false.

```
<html>
<head>
<title>Поиск элемента в массиве</title>
</head>
<body>
<?php
$a = array("первый" => 6, "второй" => 2, "третий" => 1);
if (in_array (2, $a)) echo "2 нашли!<br>";
echo "ключ найденного элемента - ".array_search(2, $a);
?>
</body>
</html>
```

РЕЗУЛЬТАТ:

2 нашли!
ключ найденного элемента - второй