### Лекция 9.

### Тема: Циклы

В *PHP* существует несколько конструкций, позволяющих выполнять повторяющиеся действия в зависимости от условия. Это циклы *while* , *do..while* , *foreach* и *for* . Рассмотрим их более подробно.

#### while

Структура:

while (выражение) { блок\_выполнения }

либо

while (выражение): блок\_выполнения endwhile;

*while* – простой цикл. Он предписывает PHP выполнять команды блока\_выполнения до тех пор, пока выражение вычисляется как True (здесь, как и в *if* , происходит приведение выражения к логическому типу). Значение выражения проверяется каждый раз в начале цикла, так что, даже если его значение изменилось в процессе выполнения блока\_выполнения, цикл не будет остановлен до конца итерации (т.е. пока все команды блока\_выполнения не будут исполнены).

<?

//эта программа напечатает все четные цифры

$i = 1;

while ($i < 10) {

if ($i % 2 == 0) print $i;

// печатаем цифру, если она четная

$i++;

// и увеличиваем $i на единицу

}

?>

Пример 3.5. Оператор while

#### do... while

Циклы *do..while* очень похожи на циклы *while* , с той лишь разницей, что истинность выражения проверяется в конце цикла, а не в начале. Благодаря этому блок\_выполнения цикла do...while гарантированно выполняется хотя бы один раз.

Структура:

do {блок\_выполнения} while (выражение);

<?

// эта программа напечатает число 12, несмотря на то

// что условие цикла не выполнено

$i = 12;

do{

if ($i % 2 == 0) print $i;

// если число четное, то печатаем его

$i++;

// увеличиваем число на единицу

}while ($i<10)

?>

Пример 3.6. Оператор do..while

#### for

Это самые сложные циклы в PHP. Они напоминают соответствующие циклы C.

Структура:

for (выражение1; выражение2; выражение3) {блок\_выполнения}

либо

for (выражение1; выражение2; выражение3): блок\_выполнения endfor;

Здесь, как мы видим, условие состоит сразу из трех выражений. Первое выражение выражение1 вычисляется безусловно один раз в начале цикла. В начале каждой итерации вычисляется выражение2. Если оно является True, то цикл продолжается и выполняются все команды блока\_выполнения. Если выражение2 вычисляется как False, то исполнение цикла останавливается. В конце каждой итерации (т.е. после выполнения всех команд блока\_выполнения ) вычисляется выражение3.

Каждое из выражений 1, 2, 3 может быть пустым. Если выражение2 является пустым, то это значит, что цикл должен выполняться неопределенное время (в этом случае PHP считает это выражение всегда истинным). Это не так бесполезно, как кажется, ведь цикл можно останавливать, используя оператор *break* .

Например, все четные цифры можно вывести с использованием цикла *for* таким образом:

<?php

for ($i=0; $i<10; $i++){

if ($i % 2 == 0) print $i;

// печатаем четные числа

}

?>

Если опустить второе выражение (условие $i<10 ), то такую же задачу можно решить, останавливая цикл оператором *break* .

<?php

for ($i=0; ; $i++){

if ($i>=10) break;

// если $i больше или равно 10,

// то прекращаем работу цикла

if ($i % 2 == 0) print $i;

// если число четное,

// то печатаем его

}

?>

Можно опустить все три выражения. В этом случае просто не будет задано начальное значение счетчика $i и оно не будет изменяться каждый раз в конце цикла. Все эти действия можно записать в виде отдельных команд либо в блоке\_выполнения, либо перед циклом:

<?php

$i=2; // задаем начальное значение счетчика

for ( ; ; ){

if ($i>=10) break;

// если $i больше или равно 10,

// то прекращаем работу цикла

if ($i % 2 == 0) print $i;

// если число четное,

// то печатаем его

$i++; // увеличиваем счетчик на единицу

}

?>

В третье выражение конструкции *for* можно записывать через запятую сразу несколько простейших команд. Например, если мы хотим просто вывести все цифры, то программу можно записать совсем просто:

<?php

for ($i=0; $i<10; print $i, $i++)

/\* Если блок\_выполнения не содержит команд

или содержит только одну команду,

фигурные скобки, в которые он заключен,

можно опускать\*/

?>

#### foreach

Еще одна полезная конструкция. Она появилась только в *PHP4* и предназначена исключительно для работы с массивами.

Синтаксис:

foreach ($array as $value) {блок\_выполнения}

либо

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

{блок\_выполнения}

В первом случае формируется цикл по всем элементам массива, заданного переменной $array. На каждом шаге цикла значение текущего элемента массива записывается в переменную $value, и внутренний счетчик массива передвигается на единицу (так что на следующем шаге будет виден следующий элемент массива). Внутри блока\_выполнения значение текущего элемента массива может быть получено с помощью переменной $value. Выполнение блока\_выполнения происходит столько раз, сколько элементов в массиве $array.

Вторая форма записи в дополнение к перечисленному выше на каждом шаге цикла записывает ключ текущего элемента массива в переменную $key, которую тоже можно использовать в блоке\_выполнения.

Когда *foreach* начинает исполнение, внутренний указатель массива автоматически устанавливается на первый элемент.

<?php

$names = array("Иван","Петр","Семен");

foreach ($names as $val) {

echo "Привет, $val <br>";

// выведет всем приветствие

}

foreach ($names as $k => $val) {

// кроме приветствия,

// выведем номера в списке, т.е. ключи

echo "Привет, $val !

Ты в списке под номером $k <br>";

}

?>

Пример 3.7. Оператор foreach