### Лекция 10

### Тема: Операторы передачи управления

Иногда требуется немедленно завершить работу *цикла* либо отдельной его итерации. Для этого используют *операторы* *break* и *continue* .

#### Break

Оператор *break* заканчивает выполнение текущего цикла, будь то *for* , *foreach* , *while* , *do..while* или *switch* . *break* может использоваться с числовым аргументом, который говорит, работу скольких управляющих структур, содержащих его, нужно завершить.

<?php

$i=1;

while ($i) {

$n = rand(1,10);

// генерируем произвольное число

// от 1 до 10

echo "$i:$n ";

// выводим номер итерации и

// сгенерированное число

if ($n==5) break;

/\* Если было сгенерировано число 5,

то прекращаем работу цикла. В этом случае

все, что находится после этой строчки

внутри цикла, не будет выполнено \*/

echo "Цикл работает <br>";

$i++;

}

echo "<br>Число итераций цикла $i ";

?>

Пример 3.8. Оператор break

Результатом работы этого скрипта будет примерно следующее:

1:7 Цикл работает

2:2 Цикл работает

3:5

Число итераций цикла 3

Если после оператора *break* указать число, то прервется именно такое количество содержащих этот *оператор циклов*. В приведенном выше примере это неактуально, поскольку *вложенных циклов* нет. Немного изменим наш скрипт:

<?php

$i=1;

while ($i) {

$n = rand(1,10);

// генерируем произвольное число

// от 1 до 10

switch ($n){

case 5:

echo "<font color=blue>

Выход из switch (n=$n)</font>";

break 1;

// прекращаем работу switch

// (первого содержащего break цикла)

case 10:

echo "<font color=red>

Выход из switch и

while (n=$n)</font>";

break 2;

// прекращаем работу switch и while

// (двух содержащих break циклов)

default:

echo "switch работает (n=$n), ";

}

echo " while работает – шаг $i <br>";

$i++;

}

echo "<br>Число итераций цикла $i ";

?>

#### continue

Иногда нужно не полностью прекратить работу цикла, а только начать его новую итерацию. Оператор *continue* позволяет пропустить дальнейшие инструкции из блока\_выполнения любого цикла и продолжить выполнение с нового круга. *continue* можно использовать с числовым аргументом, который указывает, сколько содержащих его *управляющих конструкций* должны завершить работу.

Заменим в примере предыдущего параграфа оператор *break* на *continue* . Кроме того, ограничим число шагов цикла четырьмя.

<?php

$i=1;

while ($i<=4) {

$n = rand(1,10);

// генерируем произвольное число

// от 1 до 10

echo "$i:$n ";

// выводим номер итерации и

// сгенерированное число

if ($n==5) {

echo "Новая итерация <br>";

continue;

/\* Если было сгенерировано число 5,

то начинаем новую итерацию цикла,

$i не увеличивается \*/

}

echo "Цикл работает <br>";

$i++;

}

--$i;

echo "<br>Число итераций цикла $i ";

?>

Результатом работы этого скрипта будет

1:10 Цикл работает

2:5 Новая итерация

2:1 Цикл работает

3:1 Цикл работает

4:1 Цикл работает

Число итераций цикла 4

Заметим, что после выполнения оператора *continue* работа цикла не заканчивается. В примере счетчик цикла не меняется в случае получения числа 5, поскольку он находится после оператора *continue* . Фактически с помощью *continue* мы пытаемся избежать ситуации, когда будет сгенерировано число 5. Поэтому можно было просто написать, заменив оператор *continue* на проверку истинности выражения:

<?php

$i=1;

while ($i<4) {

$n = rand(1,10);

// генерируем произвольное число

// от 1 до 10

if ($n!==5) {

echo "$i:$n <br>";

// выводим номер итерации

// и сгенерированное число

$i++;

}

}

?>

В PHP существует одна особенность использования оператора *continue* – в конструкциях *switch* он работает так же, как и *break* . Если *switch* находится внутри цикла и нужно начать новую итерацию цикла, следует использовать continue 2.