Тема: «Условные операторы и циклы »

Цель: Изучить синтаксис условных операторов и циклов. Научиться создавать сценарии с использованием условий и циклов.

Теоретическая часть

Введение

Выражения условия и циклы.

Условные выражения позволяют выполнить один из двух входящих в него блоков в зависимости от выполнения какого-либо условия. Циклы позволяют выполнить один и тот же блок несколько раз.

Условные операторы

Условные операторы являются, пожалуй, наиболее распространенными конструкциями во всех алгоритмических языках программирования. Рассмотрим основные условные операторы языка PHP.

Конструкция if

Синтаксис конструкции if :

```
<?php
if (логическое выражение) оператор;
?>
```

Согласно выражениям PHP, конструкция **if** содержит логическое выражение. Если логическое выражение истинно (**true**), то оператор, следующий за конструкцией **if** будет исполнен, а если логическое выражение ложно (**false**), то следующий за **if** оператор исполнен не будет. Приведем примеры:

```
<?php
if ($a > $b) echo "значение a больше, чем b";
?>
```

Часто вам будет необходим блок операторов, который будет выполняться при определенном условном критерии, тогда эти операторы необходимо поместить в фигурные скобки {...} Пример:

```
<?php
if ($a > $b) {
     echo "a больше b";
     $b = $a;
}
?>
```

Конструкция else

Часто возникает потребность исполнения операторов не только в теле конструкции if, если выполнено какое-либо условие конструкции if, но и в случае, если условие конструкции if не выполнено. В данной ситуации нельзя обойтись без конструкции else. В целом, такая конструкция будет называться конструкцией **if-else**:

```
if (логическое_выражение)
инструкция_1;
else
инструкция_2;
```

Если инструкция_1 или инструкция_2 должны состоять из нескольких команд, то они, как всегда, заключаются в фигурные скобки. Например:

```
<?php
if ($a > $b) {
     echo "a больше, чем b";
} else {
     echo "a НЕ больше, чем b";
}
?>
```

Иногда может быть нужно составить какое-то сложное условие, например, пользователь вводит месяц своего рождения и вам нужно проверить, что введенное число больше или равно1 и меньше либо равно 12 (так как в году 12 месяцев).
Для этого существуют операторы **and** (логическое И) и **or** (логическое ИЛИ).

```
$a = 3;
$b = -3;
if ($a>0 and $b<0)
{echo 'Верно!';}
else { echo 'Неверно!'; }
$a = 3;
if ($a>=1 and $a<=12)
{echo 'Верно!';}
else { echo 'Неверно!'; }
$a = -3;
$b = -3;
if ($a>0 or $b<0)
{echo 'Верно!';} else { echo 'Неверно!'; }
```

Операторы выбора

Операторы выбора – это фактически несколько условных операторов, объединенных в одном.
Формат выбора:

```
Switch (логическое выражение) {
case <Значение 1>:
<набор выражений 1>
[break;]
case <Значение 2>:
<набор выражений 2>
[break;]
.....
default;
<набор выражений, выполненных для остальных значений>
}
```

Пример:

```
Switch  ($a) {
  Case 1:
$out = "Единица";
Break;
Case 2:
$out = "Двойка";
Break;
Case 3:
$out = "Тройка";
Break;
Default :
$out = "Другое число";
```

В секциях **Case** используются не блоки, а наборы выражений без скобок.

Циклы

Циклы позволяют повторять определенное (и даже неопределенное - когда работа цикла зависит от условия) количество раз различные операторы. Данные операторы называются телом цикла. Проход цикла называется итерацией.
PHP поддерживает три вида циклов:

- Цикл с предусловием (**while**);
- Цикл с постусловием (**do-while**);
- Цикл со счетчиком (**for**);
- Специальный цикл перебора массивов (**foreach**).

При использовании циклов есть возможность использования операторов **break** и **continue.** Первый из них прерывает работу всего цикла, а второй - только текущей итерации.
Рассмотрим циклы PHP:

Цикл с предусловием while

Цикл с предусловием while работает по следующим принципам:

1. Вычисляется значение логического выражения.
2. Если значение истинно, выполняется тело цикла, в противном случае - переходим на следующий за циклом оператор.

Синтаксис цикла с предусловием:

```
while (логическое выражение) инструкция;
```

В данном случае телом цикла является инструкция. Обычно тело цикла состоит из большого числа операторов. Приведем пример цикла с предусловием **while**:

```
<?php
$x=0;
while ($x++<10) echo $x;
?>
```

Выводит 12345678910

Цикл с постусловием do while

В отличие от цикла while, этот цикл проверяет значение выражения не до, а после каждого прохода. Таким образом, тело цикла выполняется хотя бы один раз. Синтаксис цикла с постусловием такой:

```
do
{
тело цикла;
}
while (логическое выражение);
```

После очередной итерации проверяется, истинно ли логическое выражение, и, если это так, управление передается вновь на начало цикла, в противном случае цикл обрывается. Пример скрипта, показывающего работу цикла с постусловием **do-while**:

```
<?php
$x = 1;
do {
     echo $x;
} while ($x++<10);
?>
```

Рассмотренный сценарий выведет: 12345678910

Цикл со счетчиком for
Цикл со счетчиком используется для выполнения тела цикла определенное число раз. С помощью цикла **for** можно (и нужно) создавать конструкции, которые будут выполнять действия совсем не такие тривиальные, как простая переборка значения счетчика. Синтаксис цикла **for** такой:

```
for (инициализирующие команды; условие_цикла; команды после
итерации) { тело цикла; }
```

Цикл **for** начинает свою работу с выполнения инициализирующих команд. Данные команды выполняются только один раз. После этого проверяется условие цикла, если оно истинно (**true**), то выполняется тело цикла. После того, как будет выполнен последний оператор тела, выполняются команды после итерации. Затем снова проверяется условие цикла. Если оно истинно (**true**), выполняется тело цикла и команды после итерации, и т.д.

```
<?php
for ($x=0; $x<10; $x++) echo $x;
?>
```

Данный сценарий выводит: 0123456789

Цикл foreach
Цикл foreach используется для прохождения по всем элементам массива, их количество при этом задавать не нужно. Синтаксис такой:

```
$array = array(1, 2, 3, 4, 5);
foreach ($array - as $element)

{
echo $element.'<br>';
}
```

Вводится нова переменная **$element**, в которой будет лежать элемент массива

При каждом проходе цикла на экран будет выводиться новый элемент массива. Получится столбец элементов нашего массива $array = array(1, 2, 3, 4, 5);
Этот столбец будет выглядеть так:
1
2
3
4
5

Цикл **foreach** — очень мощная и полезная вещь, его следует использовать в том случае, если вам необходимо выполнить какие-либо действия с каждым элементом массива по отдельности, например, возвести их в квадрат:

```
$array = array(1, 2, 3, 4, 5);
foreach ($array as $element)
{
echo $element*$element;
}
```

Циклом **foreach** можно пробегать не только по обычному массиву, но и по ассоциативному. В таком случае после as следует указать такую конструкцию: **$ключ => $элемент**. В переменной **$ключ** будут хранится ключи, а в переменной **$элемент** — соответствующие этим ключам элементы:

```
$array = array('a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5);
foreach ($array as $key=>$element)
{
echo $key.$element;
}
```

Выведет: **'a1', 'b2', 'c3' и так далее...**

Если вам нужны только значения ассоциативного массива и не нужны ключи, то $ключ=> можно не писать:

```
$array = array('a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5);
foreach ($array as $element)
{
echo $element;
}
```

Выведет: '1', '2', '3' и так далее...

Практическая часть

Задание 1 (if-else)

1. Если переменная $a равна нулю, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 1, 0, -3.
2. Если переменная $a больше нуля, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 1, 0, -3.
3. Если переменная $a не равна нулю, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 1, 0, -3.

Задание 2

Создайте переменные $a и $b. Просуммируйте их, а результат запишите в переменную $result. Если $result больше 5, то присвойте переменной $result значение 5. Если же она меньше 5-ти — то умножьте ее на 10. Выведите на экран

значение переменной $result. Проверьте работу скрипта при $a и $b, равных 2 и 5, 3 и 1.

Задание 3 (OR и AND)

1. Если переменная $a больше нуля и меньше 5-ти, то выведите 'Верно!', иначе выведите 'Неверно!'. Проверьте работу скрипта при $a, равном 5, 0, -3, 2.
2. Если переменная $a равна нулю или равна двум, то поделите ее на 10, иначе прибавьте к ней 7 и выведите ее на экран. Проверьте работу скрипта при $a, равном 5, 0, -3, 2.

Задание 4 (foreach)

1. Дан массив с элементами 'html', 'css', 'php', 'js', 'jq'. С помощью цикла foreach выведите эти слова в столбик.
2. Дан массив с элементами 1, 20, 15, 17, 24, 35. С помощью цикла foreach найдите сумму элементов этого массива. Запишите ее в переменную $result.
3. Дан массив с элементами 26, 17, 136, 12, 79, 15. С помощью цикла foreach найдите сумму квадратов элементов этого массива. Результат запишите переменную $result.

Задание 5 (while и for)

1. Выведите столбец чисел от 1 до 100.
2. Выведите столбец чисел от 11 до 33.
3. Выведите столбец четных чисел в промежутке от нуля до 100.

Задание 6

Создание конструкции по определеиию возраста

1. Создайте переменную $**age**
2. Присвойте переменной **$age** произвольное числовое значение
3. Напишите конструкцию **if**, которая выводит фразу: **"Вам ещё работать и работать"** при условии, что значение переменной **$age** попадает в диапазон чисел от 18 до 59(включительно)
4. Расширьте конструкцию **if**, выводя фразу: **"Вам пора на пенсию"** при условии, что значение переменной **$age** больше 59
5. Расширьте конструкцию **if-else**, выводя фразу: "**Вам ещё рано работать"** при условии, что значение переменной **$age** попадает в диапазон чисел от 1 до 17(включительно)
6. Дополните конструкцию **if-elseif**, выводя фразу: **"Неизвестный возраст"** при условии, что значение переменной **$age** не попадает в вышеописанные диапазоны чисел