PHP. Уровень 1
Основы веб-разработки
Ветвления и функции
Занятие 2

Оператор If

При создании кода часто требуется выполнять различные действия на основе некоторого выбора. В PHP это можно делать с помощью условных операторов – оператора if, оператора if … else и оператор aelseif.

`if` – этот оператор используется для выполнения блока кода, когда выполняется условие (true).

`if...else` – этот оператор используется для выполнения блока кода, когда условие выполняется (true), или для выполнения другого блока кода, когда условие не выполняется (false).

`elseif` – комбинация if и else. Оператор расширяет оператор if, чтобы выполнялся другой оператор в случае, если исходное выражение if оценивается как FALSE. В отличие от else он будет выполнять альтернативное выражение, только если условное выражение elseif оценивается как TRUE.

В том случае, когда необходимо выполнить блок кода, если выполняется некоторое условие (true), можно использовать оператор if.

Ниже представлен синтаксис оператора if:

```
if (условие) {  // выполняемый код }
```

Строки кода оператора if заключаются в фигурные скобки ({}). Эти скобки определяют начало (открывающая скобка {) и конец (закрывающая скобка }) оператора If.

Следующий пример демонстрирует использование оператора if.

```
<?php
$number = 5;
if ($number <= 10)
{
echo "Число меньше или равно 10.";
}
?>
```

В приведенном выше примере число 5 присваивается переменной $number. Затем сценарий PHP использует оператор сравнения "<=" (меньше или равно) для сравнения значения $number с числом 10. Если значение меньше или равно 10, оператор echo выводит сообщение " Число меньше или равно 10" в окне браузера. Можно видеть, что скобки применяются для ограничения блока оператора if. Открывающая скобка { появляется сразу после оператора условия, а закрывающая скобка } — в конце оператора if.

Вспомните, что все операторы PHP должны завершаться с помощью терминатора инструкции (;): echo "Число меньше или равно 10.";

В некоторых случаях может понадобиться предоставить альтернативное сообщение. В приведенном выше примере переменная $number содержит число 15, которое больше 10. Альтернативное сообщение должно выводиться, чтобы пользователь знал: число больше 10. Это можно выполнить с помощью оператора if … else.

```
 <?php
$number = 15;

if ($number <= 10)
{
echo "Число меньше или равно 10.";
}
else
{
echo "Число больше 10";
}
?>
```

Третий тип условного оператора является структурой elseif. Оператор elseif является комбинацией if и else. Подобно else он расширяет оператор if, чтобы выполнить другой оператор, если условное выражение исходного if оценивается как FALSE. Однако в отличие от else он будет выполнять это альтернативное выражение, только если условное выражение в elseif оценивается как TRUE. В одном операторе if может быть несколько структур elseif. Первое выражение elseif (если такое имеется), которое оценивается как TRUE, будет выполнено.

```
<?php
$number = 15;
if ($number < 10)
{
	echo "Число меньше 10.";
}
elseif ($number == 10)
{
	echo "Число равно 10.";
}
else
{
	echo "Число больше 10.";
}
?>
```

В этом примере числовое значение переменной $number сравнивается с 10. Сперва оператор if проверяет, что $number меньше 10. Если этот оператор выполняется (true), выводится сообщение "Число меньше 10". Затем оператор elseif используется для проверки, что $number равно 10. Если этот оператор оценивается как true, выводится сообщение "Число равно 10". Оператор elseif выполняется, ТОЛЬКО если оператор if возвращает FALSE. Наконец, если операторы if и elseif возвращают FALSE, выполняется оператор else и выводится сообщение "Число больше 10".

Оператор switch

В дополнение к операторам if, рассмотренным в предыдущем разделе, PHP включает тип условного оператора, называемый оператором switch. Оператор switch проверяет условие. Результат этой проверки определяет, какой case выполняется. switch используется обычно, когда ищут точный (равенство) результат, вместо условия больше или меньше. При проверке диапазона значений должен применяться оператор if.

`switch` – используйте этот оператор для выбора одного из нескольких блоков кода для выполнения.

Ниже представлен синтаксис оператора `switch`.

```
<?php
switch (выражение)
{
case "значение1":
// код, который будет выполнен, если выражение = значение1;
break;

case "значение2":
// код, который будет выполнен, если выражение = значение2;
break;

default:
// код, который будет выполнен, если выражение не равно ни значение1,
ни значение2;
}
?>
```

Следующий пример демонстрирует использование оператора switch.

```
<?php
$number = 25;

switch ($number)
{
case 40:
      echo "Значение \$number равно 40";
      break;
case 25:
      echo "Значение \$number равно 25";
      break;
default:
      echo "Значение \$number отлично от 25 и 40";
}
?>
```

Функции

Функция может быть определена с помощью следующего синтаксиса:

```
function Имя_функции (параметр1, параметр2,   ... параметрN)
{
      Блок_действий
      return "значение, возвращаемое функцией";
}
```

Для вызова функции указывается ее имя и в круглых скобках список значений ее параметров, если таковые имеются

```
<?php
Имя_функции ("значение_для_параметра1",
  "значение_для_параметра2",...);
?>
```

Пример вызова функции:

```
<?php
function Summ($a,$b)
{
  $s = $a+$b;
  return $s;
}
echo Summ(3,5);
echo Summ(50,4);
?>
```

Функцию можно вызвать с любого места программы, независимо от места ее определения. Исключение составляют функции, определяемые условно (внутри условных операторов или других функций). Когда функция определяется таким образом, ее определение должно предшествовать ее вызову.

```
<?php
$make = true;
// здесь нельзя вызвать Make_event();  потому что она еще не существует,
// но можно вызвать Save_info()

Save_info("Иван","Иванов","Курс по PHP");

if ($make)
{  // определение функции Make_event()
  function Make_event()
  {
    echo "<p>Планирую изучать PHP уровень 2<br>";
  }
}
Make_event();          // теперь можно вызывать Make_event()
function Save_info($first, $last, $message)       // определение функции Save_info
{
  echo "<br>$message<br>";
  echo "Имя: ". $first . " ". $last . "<br>";
}
Save_info("Павел","Павлов", "Выбрал CSS"); // Save_info можно вызывать и здесь
?>
```

Аргументы функций

У каждой функции может быть, как мы уже говорили, список аргументов. С помощью этих аргументов в функцию передается различная информация (например, значение числа, факториал которого надо подсчитать). Каждый аргумент представляет собой переменную или константу.

С помощью аргументов данные в функцию можно передавать тремя различными способами:

- передача аргументов по значению (используется по умолчанию);
- по ссылке;
- задание значения аргументов по умолчанию.

Рассмотрим эти способы подробнее.

Когда аргумент передается в функцию по значению, изменение значения аргумента внутри функции не влияет на его значение вне функции. Чтобы позволить функции изменять ее аргументы, их нужно передавать по ссылке. Для этого в определении функции перед именем аргумента следует написать знак амперсанд "&".

```php
<?php
// напишем функцию, с передачей параметра по ссылке

function add_label(&$data_str){
  $data_str .= "(внесенное изменение)";
}
$str = "Исходная строка ";

echo $str ."<br>";     // выведет исходную строку
add_label($str);       // вызовем функцию
echo $str ."<br>";     // это выведет уже измененную строку
?>
```

В функции можно определять значения аргументов, используемые по умолчанию.

```
<?php
function Message($txt="Строка по умолчанию.")
{
// здесь параметр txt имеет по умолчанию значение "Строка по умолчанию"
  echo "Вывод текста в процедуре.<br>";
  echo $txt . "<br>";
}
Message();
    // вызываем функцию без параметра.
Message("Передаваемое в функцию сообщение.");
?>
```

Обратите внимание, что все аргументы, для которых установлены значения по умолчанию, должны находиться правее аргументов, для которых значения по умолчанию не заданы.

```
<?php //Коректное задание значений по умолчанию

function Message3($test, $txt="Строка по умолчанию.")
{
// здесь параметр txt имеет по умолчанию значение "Строка по умолчанию"
  echo "</p> Вывод текста в процедуре. </p>";
  echo $txt . "<br>".$test;
}
echo "Коректное задание значений по умолчанию </p>";

Message3("Сообщение, для которого не установлено значение по умолчанию");
    // вызываем функцию без одного параметра.

Message3("Сообщение, для которого не установлено значение по умолчанию","
Передаваемое в функцию сообщение, которое заменяет значение по
умолчанию.");

?>
```

Время жизни переменной

Временем жизни переменной называется интервал выполнения программы, в течение которого она существует. Поскольку локальные переменные имеют своей областью видимости функцию, то время жизни локальной переменной определяется временем выполнения функции, в которой она объявлена. Это означает, что в разных функциях совершенно независимо друг от друга могут использоваться переменные с одинаковыми именами. Локальная переменная при каждом вызове функции инициализируется заново.

Для того, чтобы локальная переменная сохраняла свое предыдущее значение при новых вызовах функции, ее можно объявить статической при помощи ключевого слова **static**:

Временем жизни статических переменных является время выполнения сценария. Т. е., если пользователь перезагружает страницу, что приводит к новому выполнению сценария, переменная в этом случае инициализируется заново.

```php
<?php
  function counter()
  {
    $counter = 0;
    return ++$counter;
  }
  function counterS()
  {
    static $counterS = 0;
    return ++$counterS;
  }
echo counter()."</p>";
echo counter()."</p>";
echo counterS()."</p>";
echo counterS()."</p>";
?>
```

Область видимости переменных

Переменные в функциях имеют локальную область видимости. Это означает, что если даже локальная и внешняя переменные имеют одинаковые имена, то изменение локальной переменной никак не повлияет на внешнюю переменную:

```
<?
  function sum()
  {
    $var = 5; // локальная переменная
    echo $var."</p>";
  }
  $var = 10; // глобальная переменная
  sum(); // выводит 5 (локальная переменная)
  echo("<br>$var"."</p>"); // выводит 10 (глобальная переменная)
?>
```

Глобальные переменные

Чтобы использовать внутри функции переменные, заданные вне ее, эти переменные нужно объявить как глобальные. Для этого в теле функции следует перечислить их имена после ключевого слова global:

```
<?
$a=1;
function Test_g()
{
  global $a;
  $a = $a*2;
  echo 'в результате работы функции $a='.$a."</p>";
}
echo 'вне функции $a=',$a,', ';
Test_g();
echo "<br>";
echo 'вне функции $a=',$a,', ';
Test_g();
?>
```

Возвращаемые значения

Все функции, приведенные выше в качестве примеров, выполняли какие-либо действия. Кроме подобных действий, любая функция может возвращать как результат своей работы какое-нибудь значение. Это делается с помощью утверждения return. Возвращаемое значение может быть любого типа. Когда интерпретатор встречает команду return в теле функции, он немедленно прекращает ее исполнение и переходит на ту строку, из которой была вызвана функция.

Рекурсия

Рекурсией называется такая конструкция, при которой функция вызывает саму себя. Различают прямую и косвенную рекурсии. Функция называется прямо рекурсивной, если содержит в своем теле вызов самой себя. Если же функция вызывает другую функцию, которая в свою очередь вызывает первую, то такая функция называется косвенно рекурсивной.