ЛЕКЦИЯ 5. УСЛОВНЫЕ ОПЕРАТОРЫ (ОПЕРАТОРЫ ВЕТВЛЕНИЯ)

Двумя основными операторами, обеспечивающими создание структур ветвления на основе условий, являются **if** и **switch**. Наиболее широко применяется оператор **if**, который используется в структурах перехода по условию. С другой стороны, в определенных ситуациях, особенно если приходится обеспечивать переход по одной из многочисленных ветвей в зависимости от значения единственного выражения, а применение целого ряда операторов **if** приводит к усложнению кода, более удобным становится оператор **switch**.

Прежде чем изучить эти операторы нужно разобраться в логических выражениях и операциях.

Логические операции

Логические операции позволяют комбинировать логические значения (называемые также истинностными) для получения новых логических значений. Как показано в таблице ниже, в языке PHP поддерживаются стандартные логические операции (and, or, not и xor), причем первые две имеют альтернативные версии.

Логические операции PHP	
Операция	**Описание**
and	Операция, результат которой принимает истинное значение тогда и только тогда, когда оба ее операнда имеют истинное значение

or	Операция, результат которой принимает истинное значение, если один из ее операндов (или оба операнда) имеет истинное значение
!	Операция, результат которой принимает истинное значение, если ее единственный операнд (задаваемый справа от знака операции) имеет ложное значение, и ложное значение, если операнд имеет истинное значение
xor	Операция, результат которой принимает истинное значение, если любой из ее операндов (но не оба одновременно) имеет истинное значение
&&	То же, что и операция and, но связывает свои операнды сильнее по сравнению с этой операцией
||	То же, что и операция or, но связывает свои операнды сильнее по сравнению с этой операцией

Операции && и || должны быть знакомы программистам, работающим с языком C. Операцию ! обычно называют not, поскольку она становится отрицанием для того операнда, к которому применяется.

Чтобы проверить, имеют ли оба операнда значение TRUE, следует использовать оператор AND, который можно записать и как двойной амперсанд (&&). Оба оператора, AND и &&, являются логическими, их единственное отличие в том, что оператор && имеет более высокий приоритет, чем оператор AND. То же самое относится к операторам OR и ||. Оператор AND возвращает TRUE, только если оба операнда имеют значение TRUE; в противном случае возвращается значение FALSE.

Чтобы проверить, имеет ли хотя бы один операнд значение TRUE, следует использовать оператор OR, который можно записать и как двойную вертикальную линию (||). Этот оператор возвращает TRUE, если хотя бы один из операндов имеет значение TRUE.

При использовании оператора OR в программе могут появиться трудноуловимые логические ошибки. Если PHP обнаружит, что первый операнд

имеет значение TRUE, он не станет вычислять значение второго операнда. Это позволяет экономить время исполнения, но вы должны внимательно следить за тем, чтобы код, от которого зависит корректная работа программы, не был помещен во второй операнд.

Проверить, имеет ли значение TRUE только один из операндов (но не оба сразу), позволяет оператор XOR. Этот оператор возвращает значение TRUE, если один и только один из операндов имеет значение TRUE. Если оба операнда имеют значение TRUE, оператор вернет значение FALSE.

Инвертировать логическое значение можно с помощью оператора NOT, который часто записывается и в виде восклицательного знака (!). Он возвращает значение TRUE, если операнд имеет значение FALSE, и значение FALSE, если операнд имеет значение TRUE.

В таблице ниже приведены некоторые логические выражения и их результаты:

Логические выражения и их результаты	
Пример логического выражения	**Результат**
TRUE AND TRUE	TRUE
FALSE AND TRUE	FALSE
TRUE OR FALSE	TRUE
FALSE OR FALSE	FALSE
TRUE XOR TRUE	FALSE
TRUE XOR FALSE	TRUE
!TRUE	FALSE
!FALSE	TRUE

Операции сравнения

В таблице ниже показаны операции сравнения, которые могут применяться либо с числами, либо со строками:

Операции сравнения

Операция	Наименование	Описание
==	Равно	Операция, результат которой принимает истинное значение, если ее операнды равны друг другу, в противном случае принимает ложное значение
!=	Не равно	Операция, результат которой принимает ложное значение, если ее операнды равны друг другу, в противном случае принимает истинное значение
<	Меньше	Операция, результат которой принимает истинное значение, если левый операнд меньше правого, в противном случае принимает ложное значение
>	Больше	Операция, результат которой принимает истинное значение, если левый операнд больше правого, в противном случае принимает ложное значение
<=	Меньше или равно	Операция, результат которой принимает истинное значение, если левый операнд меньше или равен правому, в противном случае принимает ложное значение
>=	Больше или равно	Операция, результат которой принимает истинное значение, если левый операнд больше или равен правому, в противном случае принимает ложное значение
===	Идентично	Операция, результат которой принимает истинное значение, если оба операнда равны друг другу и относятся к одному и тому же типу, в противном случае принимает ложное значение

Необходимо следить за тем, чтобы не допустить одну весьма распространенную ошибку — не путать операцию присваивания (=) с операцией сравнения (==).

Приоритет операций

Безусловно, не следует слишком злоупотреблять стилем программирования, в котором последовательность выполнения операций в основном обусловлена использованием правил приоритета, поскольку код, написанный в таком стиле, является сложным для восприятия теми, кто в дальнейшем будет изучать его, но следует отметить, что операции сравнения имеют более высокий приоритет, чем логические операции. Это означает, что оператор с выражением проверки, подобном приведенному ниже

```
$var1 = 14; $var2 = 15;

if (($var1 < $var2) && ($var2 < 20))
    echo '$var2 больше $var1 но меньше 20';
```

можно переписать как...

```
if ($var1 < $var2 && $var2 < 20)
   ...
```

Инструкция if - else

Инструкция if позволяет исполнить блок кода, если условное выражение в этой инструкции имеет значение TRUE; в противном случае блок кода не исполняется. В качестве условия может применяться любое выражение, включающее проверки на ненулевое значение, равенство, NULL с участием переменных и значений, возвращаемых функциями.

Не важно, какие отдельные условные выражения составляют условное предложение. Если условие истинно, исполняется программный код, заключенный в фигурные скобки ({}). В противном случае PHP игнорирует его и переходит к проверке второго условия, проверяя все условные предложения, которые вы записали, пока не наткнется на инструкцию else, после чего автоматически выполнит этот блок. Инструкция else не является обязательной.

Синтаксис инструкции if:

```
if (условное выражение)
{
    блок программного кода;
```

```
}
```

Если в результате вычисления условного выражения получается значение TRUE, то блок программного кода, расположенный после него, будет исполнен. В следующем примере если переменная $username имеет значение 'Admin', будет выведено приветственное сообщение. В противном случае ничего не произойдет:

```
$username = "Admin";

if ($username == "Admin")
{
      echo 'Добро пожаловать на страницу администратора.';
}
```

Если блок программного кода содержит только одну инструкцию, то фигурные скобки необязательны, тем не менее, хорошая привычка – ставить их всегда, поскольку с ними код легче читается и редактируется.

Необязательная инструкция else – это блок программного кода, исполняемый по умолчанию, когда условное выражение возвращает значение FALSE. Инструкцию else нельзя использовать отдельно от инструкции if, поскольку у else нет собственного условного выражения. То есть else и if в вашем коде всегда должны быть вместе:

Инструкции if и else

```
$username = "no admin";

if ($username == "Admin")
{
      echo 'Добро пожаловать на страницу администратора.';
}
else
{
      echo 'Добро пожаловать на страницу пользователя.';
}
```

Не забывайте закрывать фигурной скобкой блок кода в инструкции if, если вы поставили фигурную скобку в начале блока. В блоке else тоже должны быть открывающая и закрывающая фигурные скобки, как в блоке if.

Все это хорошо, кроме случаев, когда вам требуется проверить несколько условий подряд. Для этого подойдет инструкция elseif. Она позволяет проверять

дополнительные условия, пока не будет найдено истинное или достигнут блок else. У каждой инструкции elseif есть собственный блок кода, размещаемый непосредственно после условного выражения инструкции elseif. Инструкция elseif идет после инструкции if и перед инструкцией else, если таковая имеется.

Синтаксис инструкции elseif немного сложнее, но следующий пример поможет вам разобраться в нем:

Проверка нескольких условий

```
$username = "Guest";

if ($username == "Admin")
{
      echo 'Добро пожаловать на страницу администратора.';
}
elseif ($username == "Guest")
{
      echo 'Просмотр не доступен.';
}
else
{
      echo 'Добро пожаловать на страницу пользователя.';
}
```

Здесь проверяется два условия, и, в зависимости от значения переменной $username, выполняются разные действия. И еще есть возможность что-то сделать, если значение переменной отличается от первых двух.

Тернарный оператор ?:

Оператор ?: – это тернарный (трехместный) оператор, который принимает три операнда. Он работает аналогично инструкции if, но возвращает значение одного из двух выражений. Выражение, которое будет вычисляться, определяется условным выражением. Двоеточие (:) служит разделителем выражений:

```
(условие) ? вычислить_если_условие_истинно : вычислить_если_условие_ложно;
```

В примере ниже проверяется значение, и в зависимости от его значения (TRUE или FALSE) возвращаются разные строки:

Создание сообщения с помощью оператора ?:

```
$logged_in = TRUE;
$user = "Игорь";

$banner = (!$logged_in) ? "Зарегистрируйтесь!" : "С возвращением, $user!";
echo $banner;
```

Вполне очевидно, что приведенный выше оператор эквивалентен следующему оператору:

```
$logged_in = TRUE;
$user = "Игорь";

if (!$logged_in)
{
      $banner = "Зарегистрируйтесь!";
}
else
{
      $banner =  "С возвращением, $user!";
}
echo $banner;
```

Инструкция switch

Инструкция switch сравнивает выражение с несколькими значениями. Как правило, в качестве выражения используется переменная, в зависимости от значения которой должен быть исполнен тот или иной блок кода. Например, представим себе переменную $action, которая может иметь значения "ADD" (добавить), "MODIFY" (изменить) и "DELETE" (удалить). Инструкция switch позволяет легко определить блок кода, который должен исполняться для каждого из этих значений.

Чтобы показать разницу между инструкциями if и switch, выполним проверку переменной на соответствие нескольким значениям. В примере ниже приведен программный код, реализующий такую проверку на базе инструкции if, а в последующем примере – на базе инструкции switch:

Проверка на соответствие одному из нескольких значений (инструкция if)

```
if ($action == "ADD") {
      echo "Выполнить добавление.";
      echo "Количество инструкций в каждом блоке не ограничено.";
}
```

```
elseif ($action == "MODIFY") {
      echo "Выполнить изменение.";
}
elseif ($action == "DELETE") {
      echo "Выполнить удаление.";
}
```

Проверка на соответствие одному из нескольких значений (инструкция switch)

```
switch ($action) {
      case "ADD":
            echo "Выполнить добавление.";
            echo "Количество инструкций в каждом блоке не ограничено.";
            break;
      case "MODIFY":
            echo "Выполнить изменение.";
            break;
      case "DELETE":
            echo "Выполнить удаление.";
            break;
}
```

Инструкция switch берет значение, стоящее рядом с ключевым словом switch, и начинает сравнивать его со всеми значениями, стоящими рядом с ключевыми словамиcase, в порядке их расположения в программе. Если соответствие не найдено, не исполняется ни один из блоков. Как только совпадение обнаружено, выполняется соответствующий блок кода. Расположенные ниже блоки кода также исполняются – до конца инструкции switch или до ключевого слова break. Это удобно для организации процесса, состоящего из нескольких последовательных шагов. Если пользователь уже проделал некоторые шаги, он сможет продолжить процесс с того места, на котором прервался.

Выражение рядом с инструкцией switch должно возвращать значение элементарного типа, например число или строку. Массив можно задействовать только в виде его отдельного элемента, имеющего значение элементарного типа.

Выбор по умолчанию

Если значение условного выражения не совпало ни с одним из предложенных в инструкциях case вариантов, инструкция switch и в этом случае позволяет что-то сделать, примерно как инструкция else конструкции if, elseif, else.

Для этого нужно последним вариантом в списке выбора сделать инструкцию default:

Создание сообщения об ошибке с помощью инструкции default

```
$action = "REMOVE";
switch ($action) {
      case "ADD":
            echo "Выполнить добавление.";
            echo "Количество инструкций в каждом блоке не ограничено.";
            break;
      case "MODIFY":
            echo "Выполнить изменение.";
            break;
      case "DELETE":
            echo "Выполнить удаление.";
            break;
      default:
          echo "Ошибка: команда $action не допустима, ".
                "можно использовать только команды ADD, MODIFY и DELETE.";
}
```

Кроме обычного, инструкция switch поддерживает альтернативный синтаксис – конструкцию из ключевых слов switch/endswitch, определяющих начало и конец инструкции вместо фигурных скобок:

Инструкция switch завершается ключевым словом endswitch

```
switch ($action):
      case "ADD":
            echo "Выполнить добавление.";
            echo "Количество инструкций в каждом блоке не ограничено.";
            break;
      case "MODIFY":
            echo "Выполнить изменение.";
            break;
      case "DELETE":
            echo "Выполнить удаление.";
            break;
      default:
          echo "Ошибка: команда $action не допустима, ".
                "можно использовать только команды ADD, MODIFY и DELETE.";
endswitch;
```

Прерывание исполнения

Если должен быть исполнен только блок кода, соответствующий определенному значению, то в конце этого блока следует вставить ключевое слово break. Интерпретатор PHP, встретив ключевое слово break, перейдет к исполнению строки, расположенной после закрывающей фигурной скобки инструкции switch (или ключевого слова endswitch). Но если не использовать инструкцию break то проверка продолжается в последующих ветвях case конструкции switch. Ниже показан пример:

Что происходит при отсутствии операторов break

```
$action="ASSEMBLE ORDER";
switch ($action) {
      case "ASSEMBLE ORDER":
            echo "Собрать заказ.<br>";
      case "PACKAGE":
            echo "Упаковать.<br>";
      case "SHIP":
            echo "Доставить заказчику.<br>";
}
```

Если переменная $action будет иметь значение "ASSEMBLE ORDER", результат работы этого фрагмента будет следующим:

```
Собрать заказ.
Упаковать.
Доставить заказчику.
```

Если предположить, что стадия сборки уже была пройдена, и переменная $action имеет значение "PACKAGE", то будет получен следующий результат:

```
Упаковать.
Доставить заказчику.
```

Иногда, отсутствие операторов break бывает полезным, как в приведенном примере, где формируются стадии заказа, но в большинстве случаев следует использовать этот оператор.