МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

НАБЕРЕЖНОЧЕЛНИНСКИЙ ИНСТИТУТ

ОСНОВНЫЕ ОПЕРАТОРЫ PHP

Учебно-методическое пособие
по дисциплине
«ВЕБ-ПРОГРАММИРОВАНИЕ»

Набережные Челны
2018

Галиуллин Л.А. Основные операторы PHP: учебно-методическое пособие по дисциплине «Веб-программирование» [Электронный ресурс] / Казанский федеральный университет, Электронный архив, 2018.

Рассматриваются основные операторы PHP. Представлены выражения, иной синтаксис для оператора IF, описания классов. Приведены контрольные вопросы. Для студентов направлений подготовки «Информатика и вычислительная техника», «Программная инженерия».

Введение

PHP — это серверный язык создания сценариев (или стороны сервера), разработанный специально для Web. В HTML-страницу можно внедрить код PHP, который будет выполняться при каждом ее посещении. Код PHP интерпретируется Web-сервером и генерирует HTML или иной вывод, наблюдаемый посетителем страницы.

Разработка PHP была начата в 1994 г. и вначале выполнялась одним человеком, Расмусом Лердорфом (Rasmus Lerdorf). Этот язык был принят рядом талантливых людей и претерпел три основных редакции, пока не стал широко используемым и зрелым продуктом, с которым мы имеем дело сегодня. К январю 2001 г. он использовался почти в пяти миллионах доменов во всем мире и их число продолжает быстро расти. Количество доменов, в которых в настоящее время используется PHP, можно выяснить на странице http://www.php.net/ usage.php. PHP — это продукт с открытым исходным кодом (Open Source). У пользователя имеется доступ к исходному коду. Его можно использовать, изменять и свободно распространять другим пользователям или организациям.

Первоначально PHP являлось сокращением от Personal Home Page (Персональная начальная страница), но затем это название было изменено в соответствии с рекурсивным соглашением по наименованию GNU (GNU = Gnu's Not Unix) и теперь означает PHP Hypertext Preprocessor (Препроцессор гипертекста PHP).

В настоящее время основной версией PHP является четвертая. Адрес начальной страницы для PHP — http://www.php.net.

Выражения

Выражения - это краеугольный камень PHP. В PHP почти всё является выражениями. Простейший и наиболее точный способ определить выражение - это "что-то, имеющее значение". Простейший пример, приходящий на ум - это константы и переменные. Когда вы печатаете "$a = 5", вы присваиваете

значение '5' переменной $a.

После этого присваивания вы считаете значением $a 5, также, если вы напишете $b = $a, вы будете ожидать того же как, если бы вы написали $b = 5. Другими словами, $a это также выражение со значением 5. Запись типа '$b = ($a = 5)' похожа на запись'$a = 5; $b = 5;' (точка с запятой отмечает конец выражения). Так как присваивания рассматриваются справа налево, вы также можете написать '$b = $a = 5'.

Более сложные примеры выражений - это функции:

```
function foo () {
   return 5;
}
```

Функции - это выражения с тем значением, которое они возвращают. Так как foo() возвращает 5, значение выражение 'foo()' - 5.

PHP поддерживает 3 скалярных типа значений: целое, число с плавающей точкой и строки (скалярные выражения вы не можете "разбить" на более маленькие части, как, к примеру, массивы). PHP поддерживает 2 составных (нескалярных) типа: массивы и объекты. Каждое из таких значений может быть присвоено переменной или возвращено функцией.

PHP это язык, ориентированный на выражения, практически всё является выражениями.

Другой хороший пример выражения это префиксное и постфиксное увеличение и уменьшение. Пользователи C и многих других языков могут быть знакомы с записями variable++ and variable--. Это операторы увеличения и уменьшения. В PHP, подобно C, есть 2 типа инкремента - префиксный и постфиксный. Префиксное увеличение, которое записывается как '++$variable', приравнивается увеличенной переменной (PHP увеличивает переменную до того, как прочитать её значение).

Очень распространённый тип выражений - это выражения сравнения. Эти выражения имеют значение 0 или 1 (означает ложь или истину соответственно). PHP поддерживает > (больше, чем), >= (больше или равно), = (равно), < (меньше, чем) и <= (меньше или равно). Эти выражения в основном используются внутри условий, например оператора IF.

Последний пример выражений, с которыми мы разберёмся, это совмещённые выражения оператор-присваивание. В PHP, также как и в ряде других языков типа C, вы можете писать '$a+=3'. Значение '$a+=3' как и значение обычного присваивания - это присвоенное значение. Любой бинарный оператор может быть записан таким методом, например : '$a-=5' (вычесть 5 из значения $a), '$b*=7' (умножиить значение $a на 7) и так далее.

Есть ещё условный оператор с тремя операндами: $first ? $second : $third. Если значение первого выражения истинно (не равно 0), то исполняется второе выражение и это является результатом данного условного выражения. Иначе исполняется третий оператор.

Этот пример должен помочь Вам лучше понять выражения:

```
function double($i) /* функция удваивания переменной */
{ return $i*2; }
$b = $a = 5; /* присваиваем значения переменым $a и $b */
$c = $a++; /* постфиксное увеличение, присваиваем $c начальное значение $a (5)*/
$e = $d = ++$b; /* префиксное увеличение, $d и $e равны увеличенному значение $b */
$f = double($d++); /* $f = удвоенное значение $d до его увеличения*/
$g = double(++$e); /* $g = удвоенное значение $e после его увеличения */
$h = $g += 10; /* увеличить значение $g на 10, присвоить это значение переменной $h */
```

Не каждое выражения является оператором. Оператор имеет форму 'выражение' ';', то есть выражение, за которым следует точка с запятой. В '$b=$a=5;' $a=5 это правильное выражение, но оно не является оператором. А вот '$b=$a=5;' является оператором.

Ещё одна вещь, которую нужно упомянуть - это логические значения выражений. Во многих случаях, в основном в условных операторах и операторах циклов, вы не заинтересованы в конкретных значениях выражений, а только являются ли их значения TRUE или FALSE (в PHP нет специального типа boolean). Логические значения вычисляются примерно так же, как я в языке Perl. Любое не нулевое целое

значение - это TRUE, ноль - это FALSE. Обратите внимание на то, что отрицательные значения - это не ноль и поэтому они считаются равными TRUE. Пустая строка и строка '0' это FALSE; все остальные строки - TRUE. И насчёт составных типов (массивы и объекты) - если значение такого типа не содержит элементов, то оно считается равным FALSE; иначе подразумевается TRUE.

Далее мы будем писать 'expr' для обозначения любого правильного выражения PHP.

IF

Возможности PHP по использованию выражения IF похожи на C:

if (expr) statement

Вычисляется логический результат "expr" . Если expr равно TRUE, то PHP выполнит "statement", а если FALSE - проигнорирует.

Следующий пример выведет фразу 'a is bigger than b' если $a больше $b:

if ($a > $b) print "a is bigger than b";

Зачастую Вам требуется исполнить больше чем одно выражение по условию. Конечно, не надо окружать каждое выражение конструкцией IF. Вместо этого вы можете сгруппировать несколько выражений в блок выражений. К примеру, следующий код не только выведет фразу, но и присвоит значение $a переменной $b:

if ($a > $b) { print "a is bigger than b"; $b = $a; }

Выражение IF может иметь неограниченную степень вложенности в другие выражения IF, что позволяет Вам использовать выполнение по условию различных частей программы.

ELSE

Зачастую Вам требуется исполнить одно выражение, если соблюдается какое-либо условие и другое выражение в

противном случае. Вот для этого применяется ELSE. ELSE расширяет возможности IF по части обработки вариантов выражения, когда оно равно FALSE. Данный пример выведет фразу 'a is bigger than b' если $a больше $b, и 'a is NOT bigger than b' в противном случае:

```
if ($a > $b) { print "a is bigger than b"; }
else { print "a is NOT bigger than b"; }
```

Выражение ELSE выполняется только если выражение IF равно FALSE, а если есть конструкции ELSEIF - то если и они также равны FALSE (см. ниже).

ELSEIF

Является комбинацией IF и ELSE. ELSEIF, как и ELSE позволяет выполнить выражение, если значение IF равно FALSE, но в отличие от ELSE, оно выполнится только если выражение ELSEIF равно TRUE. К примеру, следующий код выведет 'a is bigger than b' если $a>$b, 'a is equal to b' если $a==$b, и 'a is smaller than b' если$a<$b:

```
if ($a > $b) { print "a is bigger than b";}
elseif ($a == $b) { print "a is equal to b";}
else { print "a is smaller than b"; }
```

Внутри одного выражения IF может быть несколько ELSEIF. Первое выражение ELSEIF (если таковые есть), которое будет равно TRUE, будет выполнено. В PHP вы можете написать 'else if' (два слова), что будет значить то же самое, что и 'elseif' (одно слово).

Выражение ELSEIF будет выполнено, только если выражение IF и все предыдущие ELSEIF равно FALSE, а данный ELSEIF равен TRUE.

Иной синтаксис для оператора IF: IF(): ... ENDIF;

PHP предлагает иной путь для группирования операторов с оператором IF. Наиболее часто это используется, когда вы внедряете блоки HTML внутрь оператора IF, но вообще может использоваться где угодно. Вместо использования фигурных

скобок за "IF(выражение)" должно следовать двоеточие, одно или несколько выражений и завершающий ENDIF. Рассмотрите следующий пример:

<?php if ($a==5): ?> A = 5 <?php endif; ?>

В этом примере блок "A = 5" внедрJн внутрь выражения IF, используемого альтернативным способом. Блок HTML будет виден, только если $a равно 5.

Этот альтернативный синтаксис применим и к ELSE и ELSEIF (expr) .Вот пример подобной структуры :

```
if ($a == 5): print "a equals 5"; print "...";
elseif ($a == 6): print "a equals 6"; print "!!!";
else: print "a is neither 5 nor 6";
endif;
```

WHILE

Цикл WHILE - простейший тип цикла в PHP. Он действует, как и его аналог в C. Основная форма оператора WHILE:

WHILE(expr) statement

Он предписывает PHP выполнять вложенный(е) оператор(ы) до тех пор, пока expr равно TRUE. Значение выражения проверяется каждый раз при начале цикла, так что если значение выражения изменится внутри цикла, то он не прервётся до конца текущей итерации. Если значение expr равно FALSE с самого начала, цикл не выполняется ни разу.

Как и в IF, вы можете сгруппировать несколько операторов внутри фигурных скобок или использовать альтернативный синтаксис:

WHILE(expr): выражения ... ENDWHILE;

Следующие примеры идентичны - оба выводят номера с 1 по 10:

```
/* example 1 */
$i = 1;
while ($i <= 10) { print $i++; }
/* example 2 */
$i = 1;
while ($i <= 10): print $i; $i++;
endwhile;
```

DO..WHILE

Цикл DO..WHILE очень похож на WHILE за исключением того, что значение логического выражения проверяется не до, а после окончания итерации. DO..WHILE гарантировано выполнится хотя бы один раз, что в случае WHILE не обязательно.

Для циклов DO..WHILE существует только один вид синтаксиса:

```
$i = 0; do { print $i; } while ($i>0);
```

Этот цикл выполнится один раз, так как после окончания итерации будет проверено значение логического выражения, а оно равно FALSE ($i не больше 0).

Программисты на C знакомы с иным использованием DO..WHILE, позволяющим прекратить исполнение блока операторов в середине путём внедрения его в цикл DO..WHILE(0) и использования оператора BREAK. Следующий код демонстрирует такую возможность :

```
do {
   if ($i < 5) {
      print "i is not big enough";
      break;
   }
   $i *= $factor;
   if ($i < $minimum_limit) { break; }
   print "i is ok";
} while(0);
```

FOR

Циклы FOR - наиболее мощные циклы в PHP. Они работают подобно их аналогам в C. Синтаксис цикла FOR :

FOR (expr1; expr2; expr3) statement

Первое выражение (expr1) безусловно вычисляется(выполняется) в начале цикла.

В начале каждой итерации вычисляется expr2. Если оно

равно TRUE, то цикл продолжается и выполняются вложенный(е) оператор(ы). Если оно равно FALSE, то цикл заканчивается. В конце каждой итерации вычисляется(исполняется) expr3.

Каждое из этих выражений может быть пустым. Если expr2 пусто, то цикл продолжается бесконечно (PHP по умолчанию считает его равным TRUE, как и C). Это не так бесполезно, как могло бы показаться, так как зачастую вам требуется закончить выполнение цикла, используя оператор BREAK в сочетании с логическим условием вместо использования логического выражения в FOR.

Рассмотрим следующие примеры. Все они выводят номера с 1 по 10 :

```
/* пример 1 */
for ($i = 1; $i <= 10; $i++)
   print $i;
}/* пример 2 */
for ($i = 1;;$i++) {
   if ($i > 10) break;
print $i;
}/* пример 3 */
$i = 1;
for (;;) {
   if ($i > 10) break;
print $i;
   $i++;
}/* пример 4 */
for ($i = 1; $i <= 10; print $i, $i++) ;
```

PHP также поддерживает альтернативный синтаксис FOR :

FOR (expr1; expr2; expr3): выражение; ...; endfor;

Другие языки используют оператор foreach для того, чтобы обрабатывает массивы или списки. PHP использует для этого оператор while и функции list() и each() . Для примера смотрите документацию по этим функциям.

BREAK

Прерывает выполнение текущего цикла.

```
$i = 0;
while ($i < 10) {
   if ($arr[$i] == "stop") { break; }
   $i++;
}
```

CONTINUE

CONTINUE переходит на начало ближайшего цикла.

```
while (list($key,$value) = each($arr)) {
   if ($key % 2) { // skip even members
      continue;    }
   do_something_odd ($value);
}
```

SWITCH

Оператор SWITCH похож на группу операторов IF с одинаковым выражением. Во многих случаях вам нужно сравнить переменную (или выражение) со многими различными значениями и выполнить различные фрагменты кода в зависимости от того, чему будет равно значение выражения.

Следующие 2 примера - это 2 различных пути для достижения одной цели, но один использует серию операторов IF, а другой - оператор SWITCH.

```
/* пример 1 */
if ($i == 0) { print "i equals 0"; }
if ($i == 1) { print "i equals 1"; }
if ($i == 2) { print "i equals 2"; }
/* пример 2 */
switch ($i) {
   case 0: print "i equals 0"; break;
   case 1: print "i equals 1"; break;
   case 2: print "i equals 2"; break;
}
```

SWITCH выполняет последовательно оператор за оператором. В начале код не исполняется. Только когда

встречается оператор CASE с подходящим значением, PHP начинает выполнять программу. PHP продолжает выполнять операторы до конца блока SWITCH или пока не встретит оператор BREAK. Если вы не напишете BREAK в конце цикла операторов, то PHP продолжит выполнять операторы и следующего SWITCH'a. К примеру:

```
switch ($i) {
   case 0: print "i equals 0";
   case 1: print "i equals 1";
   case 2: print "i equals 2";
}
```

В этом случае, если $i равно 0, то PHP выполнит все операторы print! Если $i равно 1, то PHP выполнит последние два print. И только если $i равно 2, вы получите ожидаемый результат и выведено будет только 'i equals 2'. Так что важно не забывать ставить BREAK. Специальный случай - это 'default case'. Этот оператор соответствует всем значениям, которые не удовлетворяют другим case'ам. К примеру :

```
switch ($i) {
   case 0: print "i equals 0"; break;
   case 1: print "i equals 1"; break;
   case 2: print "i equals 2"; break;
   default: print "i is not equal to 0, 1 or 2";
}
```

Выражения в CASE могут быть любого скалярного типа, то есть целые числа или числа с плавающей запятой, а так же строки. Массивы и объекты не будут ошибкой.

REQUIRE

Оператор REQUIRE заменяет себя содержимым указанного файла, похоже на то, как в препроцессоре C работает #include. Это означает, что вы не можете поместить require() внутрь цикла и ожидать, что он включит содержимое другого файла несколько раз в процессе каждой итерации. Для это используйте INCLUDE.

```
require ('header.inc');
```

INCLUDE

Оператор INCLUDE вставляет и выполняет содержимое указанного файла. Это случается каждый раз, когда встречается оператор INCLUDE, так что вы можете включить этот оператор внутрь цикла, чтобы включить несколько файлов :

```
$files = array ('first.inc', 'second.inc', 'third.inc');
for ($i = 0; $i < count($files); $i++) { include($files[$i]); }
```

include() отличается от require() тем, что оператор include выполняется каждый раз при его встрече, а require() заменяется на содержимое указанного файла безотносительно будет ли выполнено его содержимое или нет. Так как include() это специальный оператор, требуется заключать его в фигурные скобки при использовании внутри условного оператора.

```
/* Это неправильно и не будет работать, как хотелось бы. */
if ($condition) include($file);
else include($other);
/* А вот это - верно. */
if ($condition) { include($file);}
else { include($other); }
```

Когда файл исполняется, парсер пребывает в "режиме HTML", то есть будет выводить содержимое файла, пока не встретит первый стартовый тег PHP (<?).

FUNCTION

Функция может быть объявлена следующим образом:

```
function foo ($arg_1, $arg_2, ..., $arg_n) {
   echo "Example function.\n";
   return $retval; }
```

Внутри функции может быть любой верный код PHP, даже объявление другой функции или класса. Функции должны быть определены перед тем, как на них ссылаться.

Контрольные вопросы

1. Что Вы знаете о PHP?
2. Что Вы знаете о синтаксисе PHP?
3. Что Вы знаете о выражениях?
4. Что Вы знаете об операторе IF?
5. Что Вы знаете об операторе ELSE?
6. Что Вы знаете об операторе ELSEIF?
7. Что Вы знаете об операторе WHILE?
8. Что Вы знаете об операторе DO..WHILE?
9. Что Вы знаете об операторе FOR?
10. Что Вы знаете об операторе BREAK?

Рекомендуемые источники

1. Колдаев В.Д. Основы алгоритмизации и программирования: Учебное пособие / В.Д. Колдаев; Под ред. Л.Г. Гагариной. - М.: ИД ФОРУМ: ИНФРА-М, 2015. - 416 с. [Электронный ресурс]. http://znanium.com/bookread.php?book=336649.
2. Гагарина Л.Г. Технология разработки программного обеспечения: Учеб. пос. / Л.Г.Гагарина, Е.В.Кокорева, Б.Д.Виснадул; Под ред. проф. Л.Г.Гагариной - М.: ИД ФОРУМ: НИЦ Инфра-М, 2017. - 400 с. [Электронный ресурс]. http://znanium.com/bookread.php?book=389963.
3. Голицына О. Л. Программирование на языках высокого уровня: Учебное пособие / О.Л. Голицына, И.И. Попов. - М.: Форум, 2016. - 496 с. [Электронный ресурс]. http://znanium.com/bookread.php?book=139428.