УВАЖАЕМЫЕ СТУДЕНТЫ! Законспектируйте в своей рабочей тетради по дисциплине приведенную лекцию (объемом 4-5 страницы), ответьте *письменно* на контрольные вопросы.

Результаты работы, фотоотчет, предоставить преподавателю на e-mail: igor-gricenko-95@mail.ru в течении ТРЕХ дней.

При возникновении вопросов по приведенному материалу обращаться по следующему номеру телефона: (072)132-63-42

ВНИМАНИЕ!!! При отправке работы, не забывайте указывать ФИО студента, наименование дисциплины, дата проведения занятия (по расписанию).

Лекция № 9

Тема «Условные операторы, множественный выбор, циклы, операторы передачи управления, включения»

Операторы присваивания

В PHP так же, как и в языке C, есть базовый оператор присваивания "=" и комбинированные, только комбинированных операторов в PHP больше:

+=

-=

*=

/=

%=

.=

Пример присваивания с конкатенацией

```
$a = 2;
$a .= ' байта';
print "a='$a'<BR>";   //Напечатается: a='2 байта'
```

Оператор безусловного перехода

Оператор безусловного перехода goto доступен в PHP начиная с версии 5.3. У этого оператора сложная история, связанная с тем, что его неумелое применение может привести к трудно выявляемым ощибкам. Поэтому его нет в некоторых языках, например, в Java и JavaScript. Гонения на goto начались после

появления в 1968 году статьи Дейкстры (Edsger W. Dijkstra) "Go To Statement Considered Harmful" ("Обоснование пагубности оператора Go To"). В то время велись поиски структуры программы, устойчивой к появлению ошибок. Решения, использовать или не использовать тот или иной приём программирования, часто принимались не на основе результатов глубокого исследования, а следуя моде. Позже выяснилось, что являясь потенциально опасным, в некоторых случаях goto может сильно упростить программу

После появления блочного оператора if() ... else у меня возникла необходимость использовать goto всего два раза. В обоих случаях решались задачи с очень сложным алгоритмом, и должен признаться, что я не нашёл способа обойтись без goto, хотя теоретически это всегда возможно.

Оператор безусловного перехода служит для перехода в точку программы с указанной в нём меткой, и имеет синтаксис

```
    goto метка;
    . . .
    . . .
метка:
    . . .
```

Пример правильного применения goto

```
    . . .
    lf($c>2) goto M1
    $d = 7;
    $a = $d + 2;
    goto M2;
M1: $a = 3;
M2: print "a=$a"; //если $c>2, то напечатается: a=3
    . . .
```

Пример неправильного применения goto

```
. . .
    lf($c>2) goto M1
    $d = 7;
```

```
    goto M2;
    $a = $d + 2; //Этот оператор никогда не выполнится
M1: $a = 3;
M2: print "a=$a";
    . . .
```

Очевидно, что нельзя с помощью оператора goto входить внутрь цикла или в функции.

Операторы условия

Оператор вопросительный знак „?“ имеет вид

условие ? значение1 : значение2

Примеры

а) Вариант с присвоением результата переменной

```
$a = 3;
$b = 4;
$min = ($a < $b) ? $a : $b;
print "min=$min<BR>";// min=3
```

б) Вариант без присвоения результата переменной

```
$vozrast = 28;
$vozrast < 18 ? print 'несовершеннолетний' : print 'взрослый';
//Напечатается "взрослый"
```

Назовём **блоком** последовательность операторов, заключённую в фигурные скобки. Блок, состоящий из одного оператора в скобки можно не заключать.

Оператор if имеет вид

```
if(условие)
  блок операторов
```

Примеры

```
if($vozrast < 18) print 'несовершеннолетний';
$a = 3;
$b = 4;
```

```
if($a < $b)
{ $min = $a;
  $a = $b;
}
```

Оператор if ... else имеет вид

```
if(условие)
   блок операторов
else
   блок операторов
```

Пример

```
if($vozrast < 18)
   print 'несовершеннолетний';
else
{ $z = 40000;
  print "взрослый, зарплата $z рублей" ;
}
```

Оператор if ... elseif ... else имеет вид

```
if(условие)
   блок операторов
elseif
   блок операторов
....
elseif
   блок операторов
else
   блок операторов
```

Пример

```
/* Пусть y=f(x)
   если x < 0,      то y = -x;
```

```
  если 0 ≤ x < 2, то y = x<sup>2
  если 2 ≤ x < 5, то y = x + 2
  если      x ≥ 5, то y = 7
 Запишем эти условия на PHP
 */
 if($x < 0)
  $y = -$x;
 elseif($x < 2)
  $y = pow($x2,2);
 elseif($x < 5)
  $y = $x + 2;
 else
  $y = 7;
```

Распространено заблуждение, что в языке C есть аналогичный оператор **if ... else if ... else**. На самом деле в C есть только оператор **if ... else**. Перепишем на C предыдущий пример.

```
 if(x < 0)
  y = -x;
 else if(x < 2)
  y = pow(x,2);
 else if(x < 5)
  y = x + 2;
 else
  y = 7;
//Перепишем его ещё раз, расставляя фигурные скобки.
 if(x < 0)
  y = -x;
 else
 {  if(x < 2)
    y = pow(x,2);
  else
```

```
    {  if(x < 5)
         y = x + 2;
       else
         y = 7;
    }
  }
 // во внешние фигурные скобки заключён ОДИН составной оператор

    if(x < 2)
     y = pow(x2,2);
    else
    {  if(x < 5)
         y = x + 2;
       else
         y = 7;
    }
```

// но один, даже очень большой, оператор можно в скобки не заключать, тогда

получится составной оператор, внешне похожий на if ... elseif ... else.

Оператор switch (переключатель) какой-то странный. С одной стороны он заменяет оператор if ... elseif ... else, но только в простых случаях. и в if не используется оператор управления циклом break. С другой стороны switch даже относят к операторам цикла, но "цикл", записанный в виде оператора switch выглядит нелепо по сравнению с операторами for или while. Даже синтаксическую формулу для switch записать трудно, потому что последовательности операторов, по смыслу, являющиеся блоками, не заключены в фигурные скобки, т.е. формально блоками не являются. Поэтому синтаксис switch рассмотрим на примерах.

Примеры применения switch

//Пример 1. Похож на if ... elseif ... else

```
$a=1; $b=3; $c=4;
switch ($a+$b) // результат вычисления выражения в скобках
            // может иметь любой вид и встроенный тип
{   case 7:    //выражение в case должно быть того же типа,
            //что и в switch
      $d=2*$b;
      print "Случай \$d=6<BR>";
      break;
   case $c: //значение $a+$b совпало с $c
      $d=$a+$c;
      print "Случай \$d=5<BR>";
      break;
   case $b:
      $d=$b;
      print "Случай \$d=3<BR>";
      break;
   default :
      print"переменная \$d не определена<BR>";
}
//результат: $d=5

//Пример 2. Почти цикл. Подсчитывается сумма из четырёх единиц (6-$i при $i = 2)
   $i=2; $S=0;
  switch ($i)
{   case 1:
      $S++;
   case 2:
      $S++;
   case 3:
       $S++;
```

```
    case 4:
       $S++;
    case 5:
        $S++;
}
print "S=$S<BR>"; //S=4
```

Первому из приведённых в примере оператору switch соответствует следующий оператор if.

```
$a=1; $b=3; $c=4;
$e = $a + $b;
if($e == 7)
{  $d=2*$b;
   print "Случай \$d=6<BR>";
}
elseif ($e == $c)
{   $d=$a+$c;
   print "Случай \$d=5<BR>";
}
elseif ($e == $b)
{   $d=$b;
   print "Случай \$d=3<BR>";
}
else
   print"переменная \$d не определена<BR>";
```

Второму switch соответствует следующий оператор for.

```
$i=2; $S=0;
for($k = $i; $k < 6; $k++) $S++
```

Люди, в особенности программисты, по-разному понимают простоту и наглядность. Уважаемый читатель, если Вам нравится много раз писать слово *break*, то, пожалуйста, используйте switch вместо if. Если, как в примере 2, Вам хочется написать 11 строк вместо одной, то лучше switch вместо for.

Операторы цикла

Оператор цикла с предусловием

```
while(условие)
 блок операторов
```

Пример. Подсчитать сумму первых пяти членов натурального ряда ряда

```
$S = 0; $i = 0;
while($i $S< 5)
{ $i++;
  $S += $i;
}
print "S=$S<BR>"; //S=15
```

Оператор цикла с постусловием

```
do
 блок операторов
while(условие);
```

Пример. Подсчитать сумму первых пяти членов натурального

```
$S = 0; $i = 0;
do
{ $i++;
  $S += $i;
}
while($i < 5);
print "S=$S <BR>"; //S=15
```

В следующем примере двумя способами вычисляется квадратный корень из двух с помощью алгоритма Герона. Видна разница между операторами while и do ... while.

```
$x=2;$y=1.5;
$z = $x/$y;
while (abs($z -$y) > 0.001)
```

```
{    $y =($z + $y)/2;
     $z = $x/$y;
}
print "y=$y<BR>"; //y=1.41421568627

$x=2;$y=1.5;
do
{    $z = $x/$y;
     $y =($z + $y)/2;
}
while (abs($z -$y) > 0.001);
print "y=$y<BR>"; //y=1.41421568627
```

Оператор цикла с заданным количеством повторений

```
for(начальное_значение; условие_выхода; шаг
```

Пример. Подсчитать сумму первых пяти членов натурального ряда

```
$S=0;
for($i = 1; $i < 6; $i++)$S += $i;
print "S=$S<BR>"; //S=15
```

Управление циклом

Редко, но приходится нарушать ход выполнения цикла, например, при определённых условиях выйти из оператора цикла. Возникает необходимость в дополнительных, помимо предоставляемых операторами цикла. средствах управления циклом. Для этого в PHP служат операторы break и continue.

Оператор break служит для перехода на первый оператор после цикла

Пример. Как только элемент массива $A станет равным нулю, выйти из цикла.

```
$K = 0;
$A = Array(2,7,12, 1,11,1, 3,2,4, 9);
```

```
for($i=0; $i < 10; $i++)
{ $A[$i]--;
  if(!$A[$i]) break;
  $K++;
}
print "K = $K<BR>";//$K=3
```

Оператор continue служит для перехода на следующую итерацию цикла.

Пример. Все элементы массива уменьшаются на единицу и подсчитывается количество ненулевых элементов в обновлённом массиве.

```
$K = 0;
$A = Array(2,7,12, 1,7,5, 1,2,1, 9);
for($i=0; $i <10; $i++)
{ $A[$i]--;
  if(!$A[$i]) continue;
  $K++;
}
print "K = $K<BR>";//$K=7
```