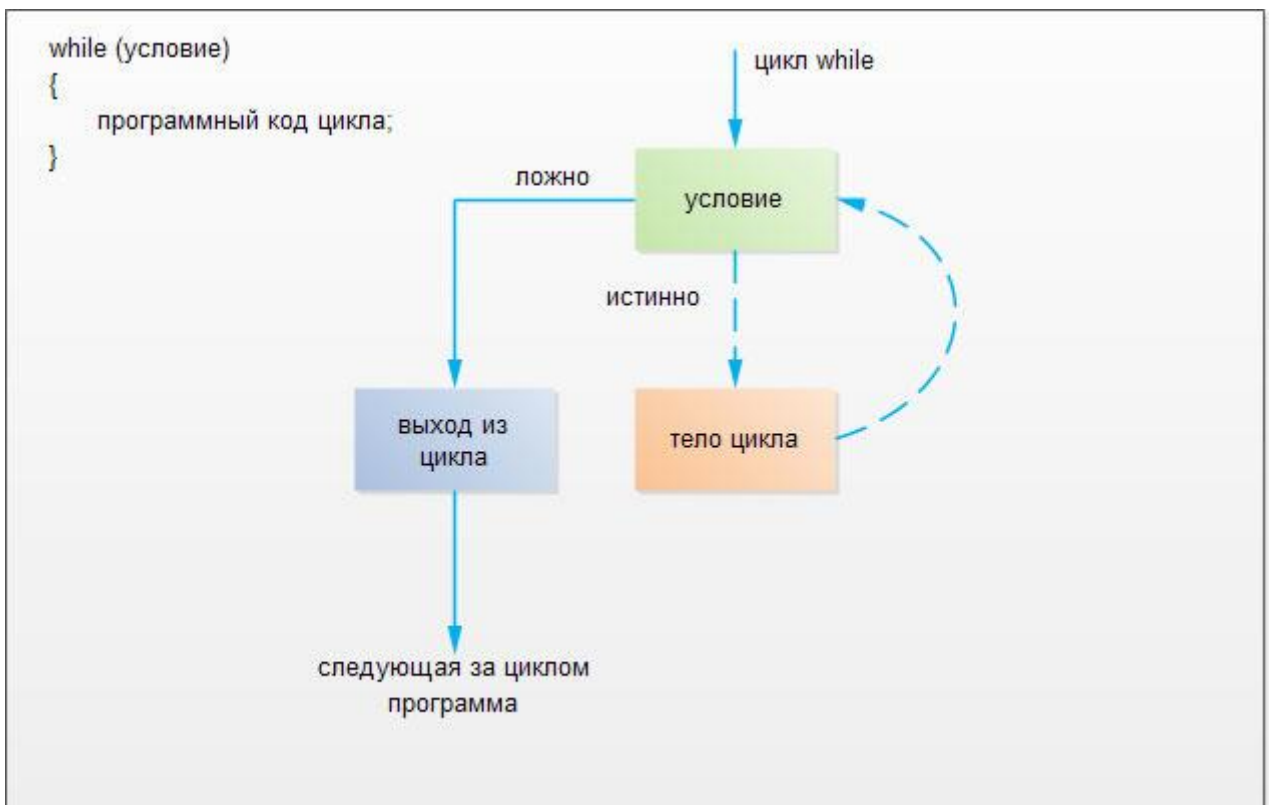


ЛЕКЦИЯ 6. ЦИКЛЫ

Цикл while	1
Структура цикла while.....	2
Простой цикл while	2
Цикл do-while	3
Цикл do-while	4
Цикл for	4
Цикл for	4
Операторы break и continue	5

Цикл while

Цикл while вычисляет условное выражение. В зависимости от результата этого вычисления выполняется следующий далее фрагмент кода. На рисунке ниже показана блок-схема данного цикла, которая иллюстрирует порядок работы цикла while:

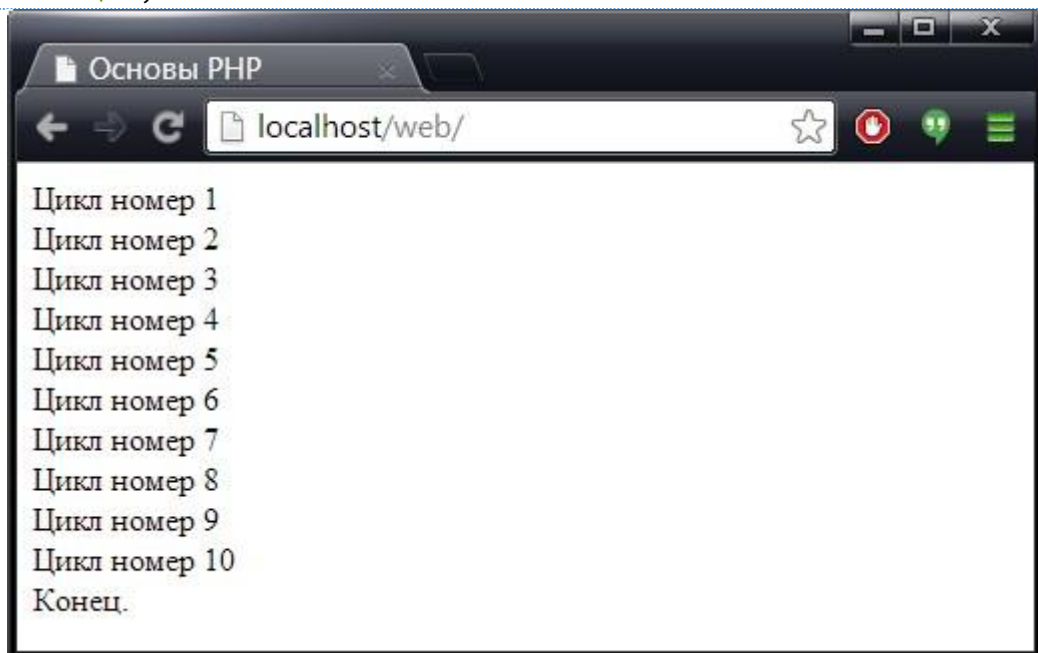


Структура цикла while

Пример ниже иллюстрирует использование цикла while:

```
$num = 1;
while ($num <= 10)
{
    print "Цикл номер $num<br>";
    $num++;
}

print 'Конец.';
```



Простой цикл while

Перед началом цикла значение переменной `$num` устанавливается равным 1. Это называется инициализацией переменной-счетчика. Каждый раз, когда выполняется блок кода, с помощью инструкции `$num++` значение переменной `$num` увеличивается на 1. После десяти итераций выражение `$num <= 10` вернет значение `FALSE`, работа цикла прекратится, и будет выведена строка "Конец".

Будьте внимательны, чтобы не создать бесконечный цикл. Неприятные последствия бесконечного цикла в сценарии: пользователь не получит запрошенную страницу, а вычислительные мощности веб-сервера будут сильно загружены.

Цикл do-while

Цикл do-while принимает условное выражение, как и цикл while, но оно помещается в конец конструкции. Синтаксис этого цикла:

```
do {  
    программный код цикла;  
}  
while(условие);
```

Данная разновидность цикла полезна, когда тело цикла должно быть исполнено хотя бы один раз, независимо от значения условного выражения. Например:

```
$num = 11;  
do  
{  
    $num--;  
    print "Цикл номер $num<br>";  
}  
while ($num <= 10 && $num > 1);  
print 'Конец.';
```

Этот фрагмент выведет:

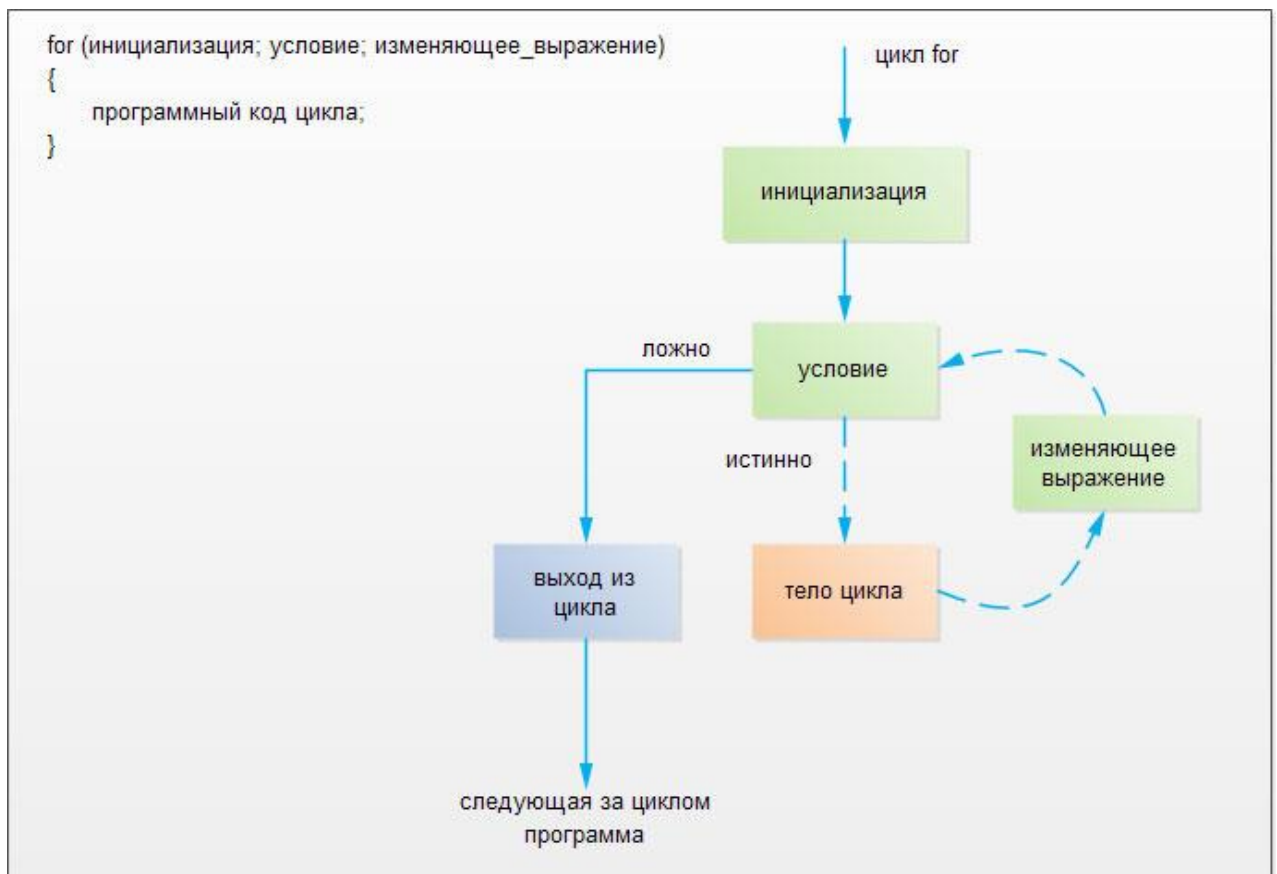


Цикл do-while

Если бы мы использовали цикл `while` с этим условием, то он не выполнялся бы ни разу, т.к. переменная $\$num = 11 > 10$. В случае использования цикла `do-while` выполняется минимум одна итерация цикла. В нашем примере в этой итерации происходит декремент переменной $\$num$ и она становится равной 10, вследствие чего выполняются последующие итерации цикла, до тех пор пока $\$num > 1$.

Цикл for

Циклы `for` в общем виде предоставляют те же функциональные возможности, что и циклы `while`, а кроме того позволяют инициализировать и изменять значение счетчика цикла. Этот цикл имеет следующий синтаксис и блок-схему:



Цикл for

Ниже показан пример использования цикла `for`, в котором достигается аналогичный результат как в примере с циклом `while`:

```
for ($num = 1; $num <= 10; $num++)
{
    print "Цикл номер $num<br>";
}

print 'Конец.';
```

Исполняя цикл `for`, PHP сначала вычисляет выражение инициализации. В каждой итерации выполняется часть кода, которая увеличивает счетчик, и затем проверяется условное выражение, чтобы узнать, не пора ли завершить цикл. В результате получается более компактная и простая для восприятия инструкция.

При определении цикла `for` можно опустить любое из трех выражений, например, выражение инициализации, но разделительные точки с запятыми (;) следует ставить всегда. Цикл `for` без выражения инициализации приведен в примере ниже:

```
$num = 1;
for ( ; $num <= 10; $num++)
{
    print "Цикл номер $num<br>";
}

print 'Конец.';
```

Кроме того, допускается включение вместо каждого отдельного выражения в цикле `for` нескольких выражений, разделенных запятыми. Условие рассматривается как истинное, если любое из его подвыражений является истинным. Таким образом, несколько подвыражений, разделенных запятыми, эквивалентны одному выражению, соединенному операциями `or`. Например:

```
for ($i = 1, $k = 1; $i < 10, $k < 10; $i++, $k++)
{
    print "$i * $k = ".$i*$k."
";
}
```

Операторы `break` и `continue`

Стандартный способ выхода из циклической конструкции состоит в создании такого условия, что проверка главного выражения цикла приводит к получению ложного значения. Еще один способ выхода из всех циклических конструкций,

включая while, do-while и for, состоит в использовании специальных команд break и continue, которые подчиняются описанным ниже правилам:

- Команда break обеспечивает выход из самой внутренней циклической конструкции, которая содержит эту команду.
- Команда continue позволяет перейти в конец текущей итерации самой внутренней циклической конструкции, которая содержит эту команду.

Например, выполнение следующего кода:

```
for ($i = 1; $i < 10; $i++)
{
    // Если значение $i является нечетным, выйти из цикла
    if ($i % 2 != 0)
        break;

    print "$i";
}
```

не выводит ничего, поскольку число 1 является нечетным и поэтому цикл for немедленно заканчивается. С другой стороны, код

```
for ($i = 1; $i < 10; $i++)
{
    // Если значение $i является нечетным, пропустить следующие операторы,
    // но выполнить новую итерацию цикла
    if ($i % 2 != 0)
        continue;

    print "$i ";
}
```

выводит строку "2 4 6 8" поскольку оператор continue позволяет пропустить все операторы, в которых были бы выведены нечетные числа.

Команда break позволяет программисту выбрать вариант организации цикла, полностью исключая необходимость проверки главного условия завершения цикла. Рассмотрим следующий код, который обеспечивает формирование и вывод списка простых чисел (т.е. чисел, которые делятся только на само это число или на единицу):

```
$limit = 500;
$number = 2;
```

```

while(TRUE) // Бесконечный цикл
{
    $test = 2;
    if ($number > $limit)
        break;

    while (TRUE)
    {
        if ($test > sqrt($number))
        {
            print "$number ";
            break;
        }

        // Проверить, делится ли $number на $test
        if ($number % $test == 0)
            break;

        $test++;
    }

    $number++;
}

```

В приведенном выше коде имеются два цикла `while`, во внешнем цикле осуществляется перебор всех чисел от 1 до 500, а во внутреннем фактически проверяется делимость с помощью каждого возможного делителя. Если во внутреннем цикле обнаруживается хотя бы один делитель, то рассматриваемое число не является простым, поэтому работа цикла прерывается без вывода какого-либо значения. Если же, с другой стороны, проверка доходит до потенциального делителя, который не меньше квадратного корня рассматриваемого числа, то можно уверенно предположить, что это число является простым, поэтому внутренний цикл прерывается после вывода данного числа. Наконец, внешний цикл прерывается после достижения значения предельного количества проверяемых чисел. В данном случае результатом является список простых чисел меньше 500:

