Работа с циклами foreach, for, while в PHP

Циклы используются для того, чтобы некоторый **участок кода выполнился несколько раз подряд**.

Зачем это нужно - представьте, что вам нужно возвести в квадрат 100 элементов массива. Если обращаться к каждому элементу отдельно по его ключу - это займет **100** строчек кода, и для того, чтобы написать этот код, нужно будет потратить довольно много времени.

Но это не нужно - у нас есть возможность сделать так, чтобы PHP выполнил за нас некоторую операцию нужное количество раз. Например, возвел все элементы массива в квадрат.

Делается это с помощью **циклов**.

Есть три вида циклов: **foreach**, **while** и **for**. Давайте разберемся, как с ними работать и чем они отличаются друг от друга.

Цикл foreach

Цикл **foreach** используется **для прохождения по всем элементам массива**.

Синтаксис такой: пишется ключевое слово **foreach**, а после него круглые скобки ( ). В этих скобках указывается переменная, в которой лежит массив, потом слово **as**, а после него - переменная, в которую при каждом проходе цикла будет ложится элемент массива.

К примеру, это может выглядеть так - **foreach($arr as $elem)**, где **$arr** - это массив, а в переменную **$elem** будут ложиться элементы массива. Имя переменной **$elem** вы придумываете сами в момент создания цикла - какое придумаете, в ту переменную и будут попадать элементы массива.

После команды **foreach()** должны идти фигурные скобки **{}**. Код, который лежит в этих скобках, называется *телом цикла*.

Этот код будет выполняться столько раз, сколько проходов сделает цикл. А он сделает столько проходов, сколько элементов у нашего массива.

Итак, синтаксис цикла **foreach** выглядит так:

```
<?php
        foreach ($arr - имя_массива as $elem – переменная_для_элемента_массива) {
                /*
                        Код, который находится между фигурными скобками будет повторяться
                        столько раз, сколько элементов у массива.
                */
        }
?>
```

Давайте решим следующую задачу: *пусть дан массив $arr с пятью элементами, выведем столбец этих элементов с помощью цикла foreach.*

Будем при каждом проходе цикла выводить на экран (с помощью echo) текущий элемент массива (тот, что лежит в переменной **$elem**), и ставить после него ter br, чтобы получался столбец элементов, а не строка:

```
<?php
        $arr = [1, 2, 3, 4, 5]; //этот массив дан
        foreach ($arr as $elem)
        /*
                Код, который находится между фигурными скобками будет повторяться
                столько раз, сколько элементов у массива (то есть 5 раз).
        */
        {
                /*
                        В переменной $elem будет лежать элемент массива,
                        каждый раз разный: сначала первый, потом второй...
                        Имя этой переменной обязательно должно отличаться от названия массива!
                */
                echo $elem.'<br>';
```

```
		/*
			При каждом проходе цикла на экран будет выводится элемент массива,
			получится столбец элементов нашего массива $arr = [1, 2, 3, 4, 5];
			Этот столбец будет выглядеть так:
			1
			2
			3
			4
			5
		*/
	}
?>
```

Цикл **foreach** - очень мощная и полезная вещь, его следует использовать в том случае, если вам необходимо выполнить какие-либо **действия с каждым элементом массива по отдельности**, например, возвести их в квадрат:

```
<?php
	//Возведем в квадрат каждый элемент следующего массива:
	$arr = [1, 2, 3, 4, 5];
	foreach ($arr as $elem) {
		/*
			В переменной $elem будет лежать элемент массива,
			каждый раз разный: сначала 1, потом 2...
		*/
		echo $elem*$elem;
	}
?>
```

Циклом **foreach** можно пробегать не только по обычному массиву, но и **по ассоциативному**. В таком случае после **as** следует указать такую конструкцию: **$ключ => $элемент**. В переменной **$ключ** будут хранится ключи, а в переменной **$элемент** - соответствующие этим ключам элементы.

Чтобы посмотреть на практике, как работать с ключами, давайте сделаем следующее - при каждом проходе цикла будем выводить на экран **ключ** массива и соответствующий ему **элемент** через дефис:

```
<?php
	//Дан ассоциативный массив $arr:
	$arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
	foreach ($arr as $key=>$elem) {
		/*
			В переменной $elem будут лежать элементы массива
			сначала '1, потом 2 и так далее,
			а в переменной $key будут лежать ключи массива
			сначала 'a', потом 'b' и так далее:
		*/
		echo $key.'-'.$elem; //выведет: 'a-1', 'b-2', 'c-3' и так далее...
	}
?>
```

Если вам нужны только значения ассоциативного массива и не нужны ключи, то **$ключ=>** можно не писать:

```php
<?php
	//Массив ассоциативный, но ключи нам не нужны:
	$arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
	foreach ($arr as $elem) {
		echo $elem; //выведет: '1', '2', '3' и так далее...
	}
?>
```

Фигурные скобки, так же, как и для if, можно не указывать - в этом случае цикл выполнит только одну строку под ним (это относится ко всем циклам, не только к foreach):

```php
<?php
	$arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
	foreach ($arr as $elem)
		echo $elem; //выведет: '1', '2', '3' и так далее...

	echo 'hello'; //выведется после цикла
?>
```

Цикл **foreach** имеет альтернативный синтаксис:

```php
<?php
	//Фигурные скобки опущены, а цикл закрывается командой endforeach:
	$arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
	foreach ($arr as $elem): //обратите внимание на двоеточие!
		echo $elem; //выведет: '1', '2', '3' и так далее...
	endforeach;
?>
```

Как и в случае с **конструкцией if-else**, мы можем **разорвать скобки PHP** внутри цикла, далее **написать что-то на HTML** и опять открыть скобки PHP – в этом случае HTML код внутри цикла повторится столько раз, сколько проходов сделает цикл (в случае foreach – это количество элементов массива):

```php
//Как работать с HTML (этого многие не знают, обратите внимание!):
<?php
	$arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
	foreach ($arr as $elem) {
		//HTML код внутри цикла повторится 5 раз:
?>
		<!--  Мы закрыли скобки PHP и теперь пишем на HTML, но в цикле!-->
		<p> {<?php echo $elem; ?>} </p>
<?php
	}
 ?>

//Альтернативный синтаксис (удобен при работе с html, так как не нужны фигурные скобки):
<?php
	$arr = ['a'=>1, 'b'=>2, 'c'=>3, 'd'=>4, 'e'=>5];
	foreach ($arr as $elem): //<--- не забываем ставить двоеточие!
?>
```

```
                <p> {<?php echo $elem; ?>} </p>
<?php
        endforeach;
?>
```

Цикл while

Цикл **while** будет выполняться до тех пор, пока **верно** (истинно) выражение, переданное ему параметром. Смотрите синтаксис:

```
<?php
        while ( пока выражение истинно ) {
                тут пишется код, который выполнится много раз;
        }
        /*
                В начале каждого прохода цикла PHP проверяет выражение в круглых
скобках:
                если оно верно - выполняет следующий проход цикла,
                а если неверно - цикл завершает свою работу.

                То есть: цикл закончится, когда выражение перестанет быть истинным.
                Если оно было ложным изначально - то он не выполнится ни разу.
        */

        //Как и для foreach есть альтернативный синтаксис:
        while ( пока выражение истинно ):
                тут пишется код, который выполнится много раз;
        endwhile;
?>
```

Давайте выведем с помощью цикла **while** столбец цифр от одного до пяти.

Для этого введем переменную **$i**, которую будем использовать для того, чтобы остановить наш цикл.

Как мы это сделаем: перед циклом поставим ей значение **1**, а внутри цикла будем при каждом проходе цикла увеличивать ее на единицу. Сначала она будет 1, потом 2, потом 3 и так далее.

В условии окончания цикла поставим **$i <= 5** - это значит, что пока **$i** будет меньше или равна пяти - цикл будет работать, а как только она станет **6** - цикл закончит свою работу.

Мы используем переменную **$i**, чтобы считать количество проходов цикла. Однако, можно также воспользоваться ей для вывода нужного нам столбца чисел от **1** до **5-ти**.

Просто при каждом проходе цикла будем выводить содержимое **$i** и тег **br** (чтобы получился столбец). Так как **$i** каждый раз увеличивается на единицу, то сначала на экран выведется 1, потом 2 и так далее до 5-ти.

Итак, вот описанный мною код:

```
<?php
        $i = 1; //начальное значение переменной $i
        while ($i <= 5) { //цикл закончится, когда $i станет больше 5-ти
                /*
                        Выводим значение переменной $i на экран
                        при каждом проходе цикла:
                */
                echo $i.'<br>';

                /*
                        С помощью оператора ++ увеличиваем $i на единицу
```

```
                при каждом проходе цикла:
            */
            $i++;
        }
?>
```

Переменная **$i**, которую мы использовали, является так называемым **счетчиком цикла**. Счетчики используются для того, чтобы выполнить цикл нужное количество раз.

Кроме того, они выполняют вспомогательную роль - в нашей задаче мы использовали счетчик, чтобы вывести цифры от 1 до 5 (и при этом для того, чтобы остановить цикл после 5-ти проходов).

Для счетчиков принято использовать буквы **i**, **j** и **k**.

Бесконечный цикл

Обратите внимание на то, что цикл **while** может выполняться **бесконечно** (но это приведет к зависанию скрипта!), достаточно передать ему выражение, которое никогда не станет ложным. Например, так:

```
<?php
        $var = true;
        while ($var === true) {
            /*
                Написанный здесь код будет выполняться 'вечно'
                (пока скрипт не будет остановлен принудительно).
                Не стоит это повторять - это приведет к зависанию сервера!
            */
        }
?>
```

Вместо **$var === true** можно написать сокращенный вариант - просто **$var** (так же, как и сокращенный вариант для if):

```
<?php
        $var = true;
        while ($var) {
            /*
                Написанный здесь код будет выполняться 'вечно'
                (пока скрипт не будет остановлен принудительно).
                Не стоит это повторять - это приведет к зависанию сервера!
            */
        }
?>
```

А можно вообще не вводить переменную **$var**, а в круглых скобках написать **true**:

```
<?php
        while (true) {
            /*
                Написанный здесь код будет выполняться 'вечно'
                (пока скрипт не будет остановлен принудительно).
                Не стоит это повторять - это приведет к зависанию сервера!
            */
        }
?>
```

Бесконечный цикл может получится не намеренно, как в примерах выше, а случайно. К примеру, я могу задать такое условие окончания цикла, которое никогда не будет достигнуто.

Посмотрите на следующий пример - в нем начальное значение **$i** равно 1, в цикле оно каждый раз увеличивается на единицу, то есть **$i** будет расти - сначала 1, потом 2, потом 3...

А условие окончания цикла такое: **$i >= 1**. То есть пока **$i** больше или равна 1 - цикл будет крутиться. В нашем случае это будет вечно (так как **$i** растет и никогда не станет меньше 1):

```
<?php
	$i = 1;
	while ($i >= 1) {
		/*
			Написанный здесь код будет выполняться 'вечно'
			(пока скрипт не будет остановлен принудительно).
			Не стоит это повторять - это приведет к зависанию сервера!
		*/

		$i++;
	}
?>
```

Цикл for

Цикл **for** является альтернативой while. Он более сложен для понимания, но чаще всего его любят больше, чем while, за то, что он занимает меньше строчек.

Его синтаксис выглядит так:

```
<?php
	for ( начальные команды; условие окончания цикла; команды после прохода цикла ) {
		тело цикла
	}
?>
```

Начальные команды - это то, что выполнится перед стартом цикла. Они выполнятся только один раз. Обычно там размещают начальные значения счетчиков, пример: **$i = 0**.

Условие окончания цикла - пока оно истинное, цикл будет работать, пример: **$i < 10**.

Команды после прохода цикла - это команды, которые будут выполнятся каждый раз при окончании прохода цикла. Обычно там увеличивают счетчики, например: **$i++**.

Давайте выведем столбец чисел от **0** до **9** с помощью цикла **for**:

```
<?php
	/*
		В начале цикла $i будет равно нулю,
		цикл будет выполнятся пока $i < 10,
		после каждого прохода к $i прибавляется единица:
	*/
	for ($i = 0; $i < 10; $i++) {
		echo $i.'<br>'; //выведет 0, 1, 2... 9
	}
?>
```

Цикл без тела

Напоминаю, что фигурные скобки в циклах можно не указывать - в этом случае цикл выполнит только одну строку под ним (не рекомендую так делать, часто приводит к ошибкам):

```
<?php
	for ($i = 0; $i < 10; $i++) //<--- точки с запятой нет
```

```
        echo $i; //выведет 0, 1, 2... 9
?>
```

А вот если после) поставить точку с запятой - цикл закроется и следующая строка в него не попадет, получится так называемый цикл без тела, который в нашем случае просто прокрутится и в результате изменит значение переменной **$i**:

```
<?php
        for ($i = 0; $i < 10; $i++); //<--- точка с запятой есть
        echo $i; //выведет 9
?>
```

Такой цикл иногда используется, вы увидите примеры его применения при разборах задач на циклы.

Несколько команд в цикле for

Если нам необходимо выполнить несколько команд в круглых скобках - указываем их через запятую:

```
<?php
        for ($i = 0, $j = 2; $i < 10; $i++, $j++,  $i = $i + $j) {


        }
?>
```

Давайте разберем приведенный цикл: до прохода цикла выполнятся две команды: **$i = 0**, **$j = 2**, а после каждой итерации - целых три: **$i++**, **$j++**, **$i = $i + $j**.

Этот пример с точки зрения программирования никакой особой пользы не несет, просто схематически показывает, что так можно делать. Запомните его, в дальнейшем это вам пригодится.

Инструкция break

Иногда нам необходимо прервать выполнение цикла досрочно, в случае с циклом foreach это значит до того, как цикл переберет все элементы массива.

Зачем такое может понадобится? Например, перед нами стоит задача выводить элементы массива до тех пор, пока не встретится число 3. Как только встретится - цикл должен завершить свою работу.

Такое можно сделать с помощью инструкции **break** - если выполнение цикла дойдет до нее, цикл закончит свою работу.

Давайте решим приведенную выше задачу - оборвем цикл, как только нам встретится число 3:

```
<?php
        $arr = [1, 2, 3, 4, 5];
        foreach ($arr as $elem) {
                if ($elem === 3)
                        break; //выходим из цикла
                else
                        echo $elem;
        }
?>
```

Инструкция continue

Существует также инструкция **continue**, при достижении которой цикл начинает новую итерацию. Иногда может быть полезна для упрощения кода, хотя практически всегда задачу можно решить и без нее.

Задачи на циклы foreach, while, for в PHP

Примеры решения задач

Задача

Задача. Дан массив с элементами **'html', 'css', 'php', 'js', 'jq'**. С помощью цикла **foreach** выведите эти слова в столбик.

Решение: Для начала необходимо создать сам массив. В данном случае мы можем воспользоваться различными способами: объявить через **[]** либо просто воспользоваться присваиванием **$arr[] = 'html'; $arr[] = 'php';** и так далее. Я выбрал **первый способ**, так как он занимает меньше места при записи:

```
<?php
	$arr = ['html', 'css', 'php', 'js', 'jq'];
?>
```

Теперь необходимо воспользоваться циклом **foreach**:

```
<?php
	$arr = ['html', 'css', 'php', 'js', 'jq'];
	foreach ($arr as $elem) {
		echo $elem.'<br>';
	}
?>
```

Как это все работает: в переменной **$elem** все элементы будут лежать по порядку: сначала **'html'**, потом **'css'** и так далее. Я буду выводить их на экран с помощью **echo**, а после вывода элемента ставить тег **
** (он делает перевод строки). Содержимое **$elem** и тег **
** являются строками, поэтому между ними я **ставлю точку** - соединяю тем самым две строки вместе.

Задача

Задача. Дан массив с элементами **10, 20, 15, 17, 24, 35**. Найдите сумму элементов этого массива. Запишите ее в переменную **$result**.

Решение: Первое решение, которое может прийти в голову новичку - это просуммировать все элементы массива 'руками': **$arr[0]+$arr[1]+...+$arr[5]**.

Однако, этот способ не правильный, так как в случае изменения массива **код придется переписывать** (например, в него добавится еще 10 элементов, или 100, что вообще будет печально). Решение необходимо делать как можно более универсальным.

Поэтому мы должны воспользоваться циклом **foreach** - в этом случае мы не будем привязаны к количеству элементов массива.

Теперь нужно подумать о том, как просуммировать элементы массива в цикле. Для этого я введу переменную **$result**, к которой при каждом проходе цикла буду прибавлять ее саму и еще один элемент массива таким образом **$result = $result + $elem**:

```
<?php
	$arr = [10, 20, 15, 17, 24, 35];

	$result = 0;
	foreach ($arr as $elem) {
		$result = $result + $elem;
	}
	echo $result; //выведем сумму на экран
?>
```

В строке **$result = 0** я зануляю переменную **$result**, если этого не сделать - в цикле возникнет ошибка в строке **$result = $result + $elem**, так как при первом проходе цикла переменная **$result** не будет определена.

Как работает строчка **$result = $result + $elem**: при первом проходе в переменной **$result** будет 0, в **$elem** – 10 (первый элемент массива), тогда в **$result** запишется 0+10 = 10.

При втором проходе цикла в **$result** лежит уже 10 (значение из прошлого прохода цикла), а в **$elem**лежит 20 (следующий элемент массива), тогда в **$result** запишется 10+20=30.

При третьем проходе цикла в **$result** лежит 30, а в **$elem** – 15, в **$result** запишется 30+15 = 45. И так далее пока цикл не закончится.

Строчку **$result = $result + $elem** можно записать короче: **$result += $elem**.

Задача

Задача. Выведите столбец чисел от 1 до 100.

Решение: Задачу можно решить как циклом **for**, так и циклом **while**. Вначале решим через цикл **while**:

```
<?php
	$i = 1; //счетчик цикла устанавливаем в 1
```

```php
        while ($i <= 100) {
                echo $i.'<br>'; //выведет 1,2... 100
                /*
                        С помощью оператора ++ увеличиваем $i
                        на единицу при каждом проходе цикла:
                */
                $i++;
        }
?>
```

Решение через цикл **for**:

```php
<?php
        /*
                В начале цикла $i будет равно 1,
                цикл будет выполнятся пока $i <= 100,
                после каждого прохода к $i прибавляется единица:
        */
        for ($i = 1; $i <= 100; $i++) {
                echo $i.'<br>'; //выведет 1,2... 100
        }
?>
```

Задачи для решения

Работа с foreach

1. Дан массив с элементами **'html'**, **'css'**, **'php'**, **'js'**, **'jq'**. С помощью цикла **foreach** выведите эти слова в столбик.

2. Дан массив с элементами **1**, **2**, **3**, **4**, **5**. С помощью цикла **foreach** найдите **сумму** элементов этого массива. Запишите ее в переменную **$result**.

3. Дан массив с элементами **1**, **2**, **3**, **4**, **5**. С помощью цикла **foreach** найдите **сумму квадратов**элементов этого массива. Результат запишите переменную **$result**.

Работа с ключами

4. Дан массив **$arr**. С помощью цикла **foreach** выведите на экран столбец ключей и элементов в формате 'green - зеленый'.

```php
$arr = ['green'=>'зеленый', 'red'=>'красный','blue'=>'голубой'];
```

5. Дан массив **$arr** с ключами **'Коля'**, **'Вася'**, **'Петя'** и с элементами **'200'**, **'300'**, **'400'**. С помощью цикла **foreach** выведите на экран столбец строк такого формата: *'Коля - зарплата 200 долларов.'*.

Циклы while и for

Решите эти задачи сначала через цикл while, а затем через цикл for.

6. Выведите столбец чисел от **1** до **100**.

7. Выведите столбец чисел от **11** до **33**.

8. Выведите столбец четных чисел в промежутке от **0** до **100**.

9. С помощью цикла найдите сумму чисел от **1** до **100**.

Задачи

10. Дан массив с элементами **2**, **5**, **9**, **15**, **0**, **4**. С помощью цикла **foreach** и оператора **if** выведите на экран столбец тех элементов массива, которые больше 3-х, но меньше 10.

11. Дан массив с числами. Числа могут быть положительными и отрицательными. Найдите сумму положительных элементов этого массива.

12. Дан массив с элементами **1, 2, 5, 9, 4, 13, 4, 10**. С помощью цикла **foreach** и оператора **if**проверьте есть ли в массиве элемент со значением, равным 4. Если есть - выведите на экран 'Есть!' и выйдите из цикла. Если нет - ничего делать не надо.

13. Дан массив числами, например: ['10', '20', '30', '50', '235', '3000']. Выведите на экран только те числа из массива, которые начинаются на цифру **1**, **2** или **5**.

14. Дан массив с элементами **1, 2, 3, 4, 5, 6, 7, 8, 9**. С помощью цикла **foreach** создайте строку **'-1-2-3-4-5-6-7-8-9-'**.

15. Составьте массив дней недели. С помощью цикла **foreach** выведите все дни недели, а выходные дни выведите жирным.

16. Составьте массив дней недели. С помощью цикла **foreach** выведите все дни недели, а текущий день выведите *курсивом*. Текущий день должен храниться в переменной **$day**.

Задачи посложнее

17. С помощью цикла **for** заполните массив числами от 1 до 100. То есть у вас должен получится массив **[1, 2, 3... 100]**.

18. Дан массив **$arr**. С помощью цикла **foreach** запишите английские названия в массив **$en**, а русские - в массив **$ru**.

```
$arr = ['green'=>'зеленый', 'red'=>'красный', 'blue'=>'голубой'];
```

19. Дано число **$num=1000**. Делите его на 2 столько раз, пока результат деления не станет меньше 50. Какое число получится? Посчитайте количество итераций, необходимых для этого (*итерация* - это проход цикла). Решите задачу сначала через цикл while, а потом через цикл for.