

## Что такое массив PHP?

Массив PHP - это переменная, которая хранит более одной части связанных данных в одной переменной.

Думайте о массиве как о коробке конфет с щелями внутри.

Блок представляет собой сам массив, а пробелы, содержащие шоколадные конфеты, представляют значения, хранящиеся в массивах.

Диаграмма ниже иллюстрирует приведенный выше синтаксис.

## Числовые массивы

Числовые массивы используют число в качестве ключей доступа.

Ключ доступа - это ссылка на слот памяти в переменной массива.

Ключ доступа используется всякий раз, когда мы хотим прочитать или присвоить новое значение элементу массива.

Ниже приведен синтаксис для создания числового массива в php. Пример массива

```
<?php
$variable_name[n] = value;
?>

Или

<?php
$variable_name = array(n => value, ...);
?>
```

- “\$variable\_name...” - это имя переменной
- «[N]» - индекс доступа к элементу
- «value» - это значение, присвоенное элементу массива.

Давайте теперь посмотрим на пример числового массива.

Предположим, у нас есть 5 фильмов, которые мы хотим сохранить в переменных массива.

Мы можем использовать приведенный ниже пример, чтобы сделать это.

```
<? PHP

$movie[0] = 'Shaolin Monk';
$movie[1] = 'Drunken Master';
$movie[2] = 'American Ninja';
$movie[3] = 'Once upon a time in China';
$movie[4] = 'Replacement Killers';?>
```

```
$movie[0] = 'Shaolin Monk';  
$movie[1] = 'Drunken Master';  
$movie[2] = 'American Ninja';  
$movie[3] = 'Once upon a time in China';  
$movie[4] = 'Replacement Killers';
```

*Numeric numbers used as element  
access keys*

Каждому фильму присваивается порядковый номер, который используется для извлечения или изменения его значения. Соблюдайте следующий код:

```
<? PHP
```

```
$movie[0]="Shaolin Monk";  
$movie[1]="Drunken Master";  
$movie[2]="American Ninja";  
$movie[3]="Once upon a time in China";  
$movie[4]="Replacement Killers";  
echo $movie[3];  
$movie[3] = " Eastern Condors";  
echo $movie[3];?>
```

**Выход:**

```
Once upon a time in China Eastern Condors
```

Как видно из приведенных выше примеров, работа с массивами в PHP при работе с несколькими значениями одинаковой природы очень проста и гибка.

В качестве альтернативы вышеупомянутые переменные массива также могут быть созданы с использованием следующего кода.

```
<? PHP
```

```
$movie = array(0 => "Shaolin Monk",  
              1 => "Drunken Master",  
              2 => "American Ninja",  
              3 => "Once upon a time in China",  
              4 => "Replacement Killers" );  
echo $movie[4];?>
```

Выход:

```
Replacement Killers
```

## PHP Ассоциативный Массив

Ассоциативный массив отличается от числового массива в том смысле, что ассоциативные массивы используют описательные имена для ключей идентификаторов.

Ниже приведен синтаксис для создания ассоциативного массива в php.

```
<? PHP

$variable_name['key_name'] = value;

$variable_name = array('keyname' => value);?>
```

ВОТ,

- «\$ Variable\_name...» - это имя переменной
- «['Key\_name']» - индекс доступа к элементу
- «value» - это значение, присвоенное элементу массива.

Давайте предположим, что у нас есть группа людей, и мы хотим назначить пол каждого человека по имени.

Для этого мы можем использовать ассоциативный массив. Приведенный ниже код помогает нам в этом.

```
<? PHP

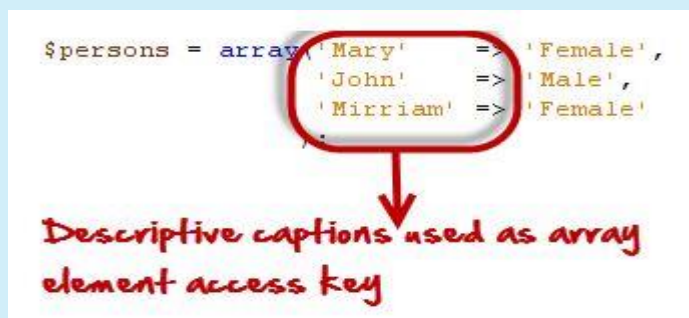
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");

print_r($persons);

echo "";

echo "Mary is a " . $persons["Mary"];?>
```

ВОТ,



```
$persons = array('Mary' => 'Female',
                 'John' => 'Male',
                 'Mirriam' => 'Female');
```

Descriptive captions used as array element access key

Выход:

```
Array ( [Mary] => Female [John] => Male [Mirriam] => Female ) Mary is a Female
```

Ассоциативный массив также очень полезен при извлечении данных из базы данных.

Имена полей используются в качестве ключей идентификатора.

## PHP Многомерные массивы

Это массивы, которые содержат другие вложенные массивы.

Преимущество многомерных массивов состоит в том, что они позволяют группировать связанные данные вместе.

Давайте теперь посмотрим на практический пример, который реализует многомерный массив php.

В таблице ниже приведен список фильмов по категориям.

Название фильма	категория
Pink Panther	Comedy
John English	Comedy
Die Hard	Action
Expendables	Action
The Lord of the rings	Epic
Romeo and Juliet	Romance
See no evil hear no evil	Comedy

Приведенная выше информация может быть представлена в виде многомерного массива.

Код ниже показывает реализацию.

```
<? PHP

$movies =array(

"comedy" => array("Pink Panther", "John English", "See no evil hear no evil"),

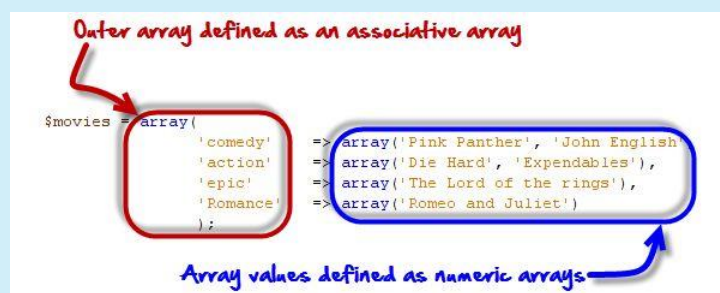
"action" => array("Die Hard", "Expendables"),

"epic" => array("The Lord of the rings"),

"Romance" => array("Romeo and Juliet")

);

print_r($movies);?>
```



### Выход:

```
Array ( [comedy] => Array ( [0] => Pink Panther [1] => John English [2] => See no evil hear no evil ) [action] => Array ( [0] => Die Hard [1] => Expendables ) [epic] => Array ( [0] => The Lord of the rings ) [Romance] => Array ( [0] => Romeo and Juliet ) )
```

Другой способ определить тот же массив заключается в следующем

```
<?php
$film=array(
    "comedy" => array(
        0 => "Pink Panther",
        1 => "john English",
        2 => "See no evil hear no evil"
    ),
    "action" => array (
        0 => "Die Hard",
        1 => "Expendables"
    ),
    "epic" => array (
        0 => "The Lord of the rings"
    ),
    "Romance" => array
        (
            0 => "Romeo and Juliet"
        )
);
echo $film["comedy"][0];
?>
```

### Выход:

Pink Panther

Примечание: числовой массив фильмов был вложен в ассоциативный массив категорий

## Массивы PHP: операторы

оператор	название	Описание	Как это сделать	Выход
$x + y$	<b>Union</b>	Объединяет элементы из обоих массивов	<pre>&lt;? PHP  \$ x = array ('id' =&gt; 1); \$ y = array ('value' =&gt; 10); \$ z = \$ x + \$ y;  ?&gt;</pre>	Array ([id] => 1 [value] => 10)
$x == y$	<b>Equal</b>	Сравнивает два массива, если они равны, и возвращает true, если да.	<pre>&lt;? PHP  \$ x = array ("id" =&gt; 1); \$ y = array ("id" =&gt; "1"); if (\$ x == \$ y) { echo "true"; } else { echo "false"; }  ?&gt;</pre>	Правда или 1
$x === y$	<b>Identical</b>	Сравнивает значения и типы данных	<pre>&lt;? PHP  \$ x = array ("id" =&gt; 1); \$ y = array ("id" =&gt; "1"); if (\$ x === \$ y) { echo "true"; } else { echo "false"; }  ?&gt;</pre>	Ложь или 0
$x != y, x <> y$	<b>Not equal</b>		<pre>&lt;? PHP  \$ x = array ("id" =&gt; 1); \$ y = array ("id" =&gt; "1"); if (\$ x! = \$ y) { эхо "правда"; } else { эхо "ложь"; }  ?&gt;</pre>	Ложь или 0

оператор	название	Описание	Как это сделать	Выход
<code>x! == y</code>	<b>Non identical</b>		<pre>&lt;? PHP \$ x = array ("id" =&gt; 1); \$ y = array ("id" =&gt; "1"); if (\$ x! == \$ y) { echo "true"; } else { echo "false"; } ?&gt;</pre>	Правда или 1

## PHP Array Функции

### Функция подсчета

Функция count используется для подсчета количества элементов в массиве php. Код ниже показывает реализацию.

```
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo count($lecturers);
?>
```

**Выход:**

3

### функция is\_array

Функция is\_array используется для определения, является ли переменная массивом или нет. Давайте теперь посмотрим на пример, который реализует функции is\_array.

```
<?php
$lecturers = array("Mr. Jones", "Mr. Banda", "Mrs. Smith");
echo is_array($lecturers);
?>
```

**Выход:**

1

### Сортировать

Эта функция используется для сортировки массивов по значениям.

Если значения буквенно-цифровые, они сортируются в алфавитном порядке.

Если значения являются числовыми, они сортируются в порядке возрастания.

Он удаляет существующие ключи доступа и добавляет новые числовые ключи.

Выход этой функции представляет собой числовой массив

```
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
sort($persons);
print_r($persons);
?>
```

**Выход:**

```
Array ( [0] => Female [1] => Female [2] => Male )
```

### **ksort**

Эта функция используется для сортировки массива по ключу. Следующий пример иллюстрирует его использование.

```
<? PHP
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
ksort($persons);
print_r($persons);?>
```

**Выход:**

```
Array ( [John] => Male [Mary] => Female [Mirriam] => Female )
```

### **asort**

Эта функция используется для сортировки массива по значениям. Следующий пример иллюстрирует его использование.

```
<?php
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");
asort($persons);
print_r($persons);
?>
```

**Выход:**

```
Array ( [Mary] => Female [Mirriam] => Female [John] => Male )
```



## Зачем использовать массивы?

- Содержимое массивов можно растянуть,
- Массивы легко помогают группировать связанную информацию, такую как данные для входа на сервер, вместе
- Массивы помогают писать более чистый код.

## Резюме

- Массивы - это специальные переменные с возможностью хранения нескольких значений.
- Массивы являются гибкими и могут легко растягиваться, чтобы вместить больше значений.
- Числовые массивы используют числа для ключей массива
- PHP Ассоциативный массив использует описательные имена для ключей массива
- Многомерные массивы содержат внутри себя другие массивы.
- Функция `count` используется для получения количества элементов, которые были сохранены в массиве
- Функция `is_array` используется для определения, является ли переменная допустимым массивом или нет.
- Другие функции массива включают сортировку, ксорт, сортировку и т. Д.

## Управляющие конструкции PHP: If else, Switch Case

### Что такое Управляющие конструкции?

Выполнение кода можно сгруппировать по категориям, как показано ниже

- **Последовательный** - этот включает в себя выполнение всех кодов в том порядке, в котором они были написаны.
- **Решение** - это включает в себя выбор с учетом ряда вариантов. Выполненный код зависит от значения условия.

Управляющие конструкции - это блок кода, который определяет путь выполнения программы в зависимости от значения заданного условия.

Давайте теперь посмотрим на некоторые структуры управления, которые поддерживает PHP.

### PHP IF ELSE

IF ... then ... else самая **простая структура управления** . Он оценивает условия, используя булеву логику. **Когда использовать if... then... else**

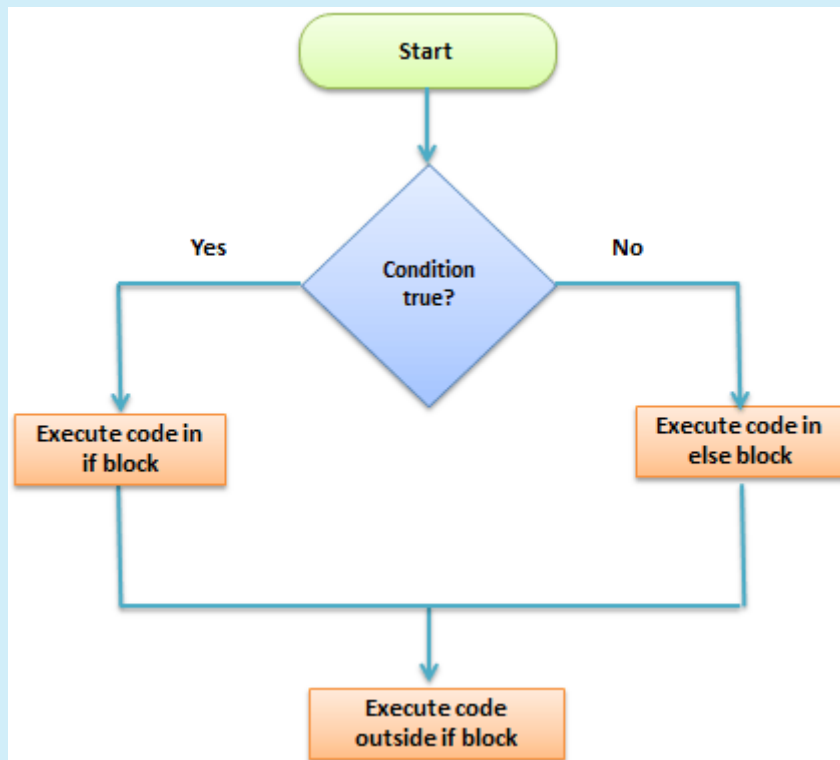
- У вас есть блок кода, который должен быть выполнен, только если выполняется определенное условие
- У вас есть два варианта, и вы должны выбрать один.
- If ... then ... else, if ... используется, когда вам нужно выбрать более двух вариантов, и вам нужно выбрать один или несколько

**Синтаксис** Синтаксис if... then... else ;

```
<?php
if (condition is true) {
block one }
else {
block two
}
?>
```

- « **If (условие истинно)**» является управляющей структурой
- « **block one**» - это код, который будет выполнен, если условие истинно
- **{... Else...}** - это запасной вариант, если условие ложно
- « **block two** » - блок кода, выполняемый, если условие ложно

**Как это работает** Блок-схема, показанная ниже, иллюстрирует, как работает структура управления if then... else



**Давайте посмотрим на это в действии.** Приведенный ниже код использует «if... then... else» для определения большего значения между двумя числами.

```
<?php
$first_number = 7;
$second_number = 21;
if ($first_number > $second_number){
echo "$first_number is greater than $second_number";
}else{
echo "$second_number is greater than $first_number";
}
?>
```

**Выход:**

21 is greater than 7

## PHP Switch Case

Переключатель... **case** аналогичен структуре управления **if then... else** .

Он **выполняет** только один блок кода в зависимости от **значения** условия.

Если условия не были выполнены, выполняется блок кода по умолчанию.

Он имеет следующий основной синтаксис.

```
<?php

switch(condition){

case value:

    //block of code to be executed

    break;

case value2:

    //block of code to be executed

    break;

default:

    //default block code

    break;

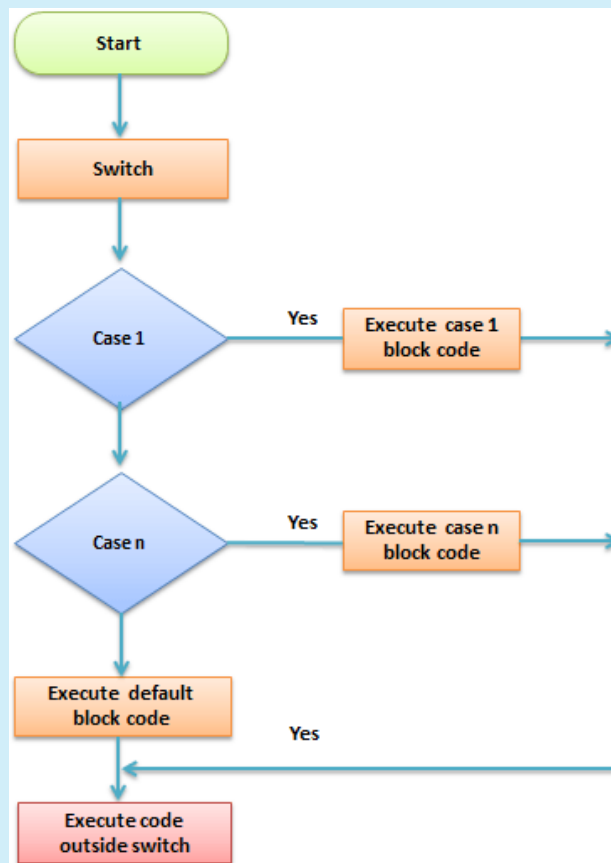
}

?>
```

- «**Switch (...) {...}**» - код блока структуры управления
- «**Case value: case...**» - блоки кода, которые должны быть выполнены в зависимости от значения условия
- «**Default:**» - блок кода, который будет выполнен, когда никакое значение не соответствует условию

## Как это работает

Блок-схема, показанная ниже, иллюстрирует, как работает структура управления коммутатором.



## Практический пример

Приведенный ниже код использует структуру управления коммутатором для отображения сообщения в зависимости от дня недели.

```
<?php
$today = "wednesday";
switch($today){
case "sunday":
echo "pray for us sinners."; break;
case "wednesday":
echo "ladies night, take her out for dinner"; break;
case "saturday":
echo "take care as you go out tonight."; break;
default:
echo "have a nice day at work"; break;
}?>
```

## Выход:

ladies night, take her out for dinner

## Резюме

- Управляющие структуры используются для контроля выполнения программы
- Если тогда ... то еще, когда у вас есть больше, чем маршрутный блок кода для выполнения в зависимости от значения условия
- Переключатель... регистр используется, когда у вас есть несколько кодов блоков, и вам нужно выполнить только один из них в зависимости от значения установленного регистра.

## PHP Loop: For, ForEach, while, Do While

Цикл - это итеративная структура управления, которая включает выполнение одного и того же количества кода несколько раз, пока не будет выполнено определенное условие.

### PHP для цикла

Вышеприведенный код выводит «21 больше, чем 7». Для циклов For ... циклы выполняют блок кода указанное число раз. Есть в основном два типа циклов for;

- для
- для каждого.

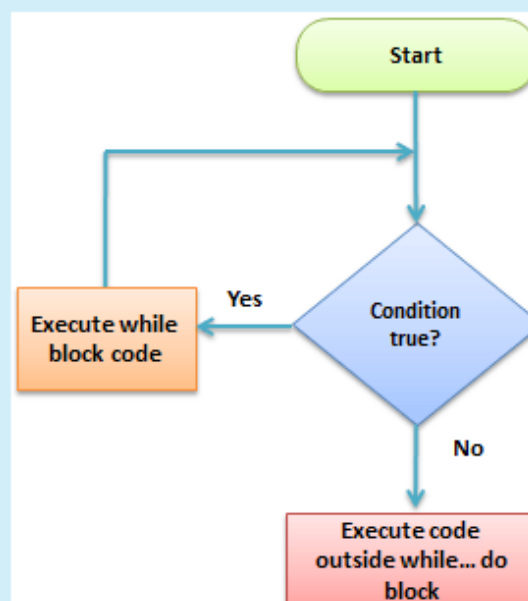
Давайте теперь посмотрим на них отдельно. **Цикл For** имеет следующий основной **синтаксис**

```
<?php
for (initialize; condition; increment){
//code to be executed
}
?>
```

- «for...{...}» - блок цикла
- « **initialize** » обычно целое число; используется для установки начального значения счетчика.
- «**condition**» условие, которое оценивается для каждого выполнения php. Если значение равно true, выполнение цикла for ... прекращается. Если значение равно false, выполнение цикла for ... продолжается.
- «**increment**» используется для увеличения начального значения целого числа счетчика.

### Как это работает

Блок-схема, показанная ниже, иллюстрирует, как работает цикл for в php.



## Как кодировать

В приведенном ниже коде используется цикл for... для вывода значений, умноженных с 10 на 0 до 10

```
<?php
for ($i = 0; $i < 10; $i++){
    $product = 10 * $i;
    echo "The product of 10 * $i is $product <br/>";
}
?>
```

## Выход:

```
The product of 10 x 0 is 0
The product of 10 x 1 is 10
The product of 10 x 2 is 20
The product of 10 x 3 is 30
The product of 10 x 4 is 40
The product of 10 x 5 is 50
The product of 10 x 6 is 60
The product of 10 x 7 is 70
The product of 10 x 8 is 80
The product of 10 x 9 is 90
```

## PHP для каждого цикла

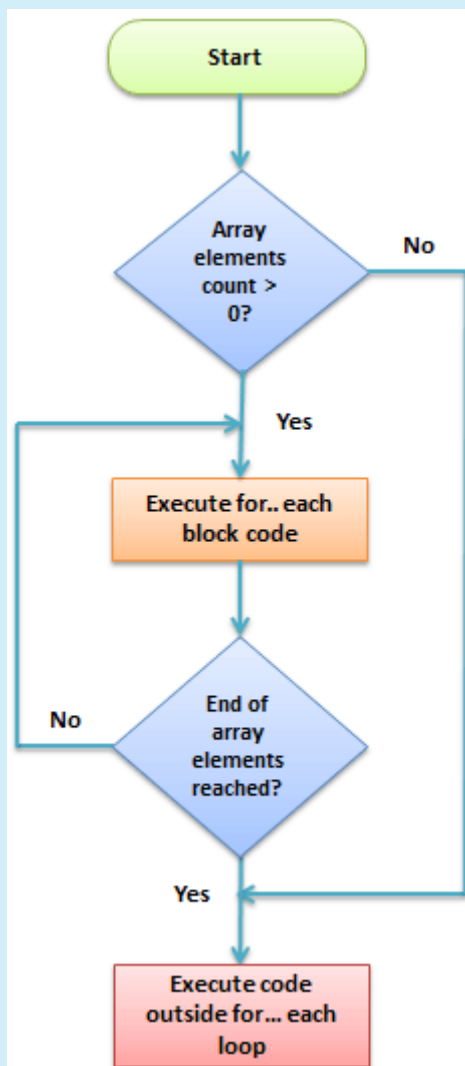
Цикл php foreach используется для перебора значений массива. Имеет следующий основной синтаксис

```
<?php
foreach($array_variable as $array_values){
    block of code to be executed
}
?>
```



- «Foreach (...) {...}» - код блока цикла foreach php
- «\$ Array\_data» - переменная массива, которая будет проходить через
- «\$ Array\_value» - временная переменная, которая содержит текущие значения элемента массива.
- «block of code ...» - это фрагмент кода, который работает со значениями массива

**Как это работает** На приведенной ниже блок-схеме показано, как работает цикл for... each...



### Практические примеры

Приведенный ниже код использует для ... каждого цикла для чтения и печати элементов массива.

```

<?php
$animals_list = array("Lion","Wolf","Dog","Leopard","Tiger");
foreach($animals_list as $array_values){
echo $array_values . "<br>";
}
?>

```

### Выход:

Lion  
Wolf  
Dog  
Leopard  
Tiger

Давайте посмотрим на другой пример, который проходит через **ассоциативный массив** .

Ассоциативный массив использует буквенно-цифровые слова для ключей доступа.

```
<?php  
  
$persons = array("Mary" => "Female", "John" => "Male", "Mirriam" => "Female");  
  
foreach($persons as $key => $value){  
  
echo "$key is $value". "<br>";  
  
}  
  
?>
```

Имена использовались как ключи массива, а пол - как значения.

### Выход:

Mary is Female  
John is Male  
Mirriam is Female

## While цикл

### PHP while цикл

Они используются для многократного выполнения блока кода, пока не будет выполнено заданное условие

### Когда использовать циклы while

- Циклы while используются для выполнения блока кода, пока определенное условие не станет истинным.
- Вы можете использовать цикл while для чтения записей, возвращаемых из запроса к базе данных.

### Типы циклов

- **Do ... while** - выполнить блок кода хотя бы один раз, прежде чем оценивать условие
- **Пока ...** - сначала проверяет состояние. Если он принимает значение true, блок кода выполняется до тех пор, пока условие выполняется. Если значение равно false, выполнение цикла while прекращается.

## While цикл

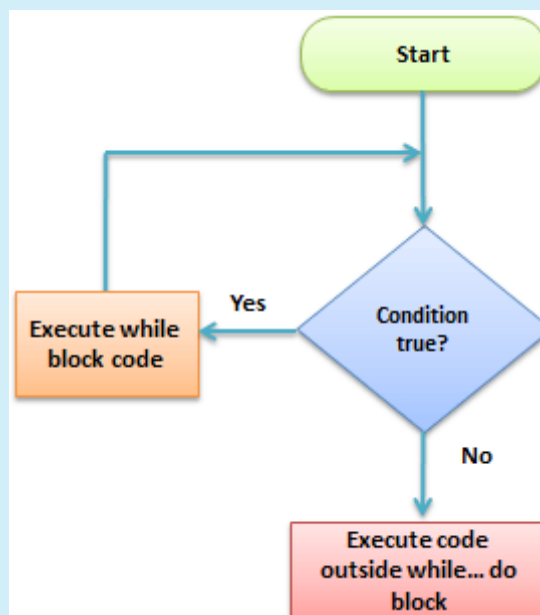
Имеет следующий синтаксис

```
<?php  
while (condition){  
    block of code to be executed;  
}  
?>
```

- «**While (...)** {...}» - это код блока цикла while
- «**condition**» - это условие, которое будет оцениваться циклом while
- «**block of code...**» - это код, который будет выполнен, если условие выполнено

### Как это работает

Блок-схема, показанная ниже, показывает, как работает цикл while ...



### Практический пример

Приведенный ниже код использует цикл while... для печати чисел от 1 до 5.

```
<?php  
$i = 0;  
while ($i < 5){  
    echo $i + 1 . "<br>";  
    $i++;  
}  
?>
```

## Выход:

1  
2  
3  
4  
5

## PHP do while

Разница между циклами while... и do..., в то время как цикл do... while выполняется хотя бы один раз перед оценкой условия.

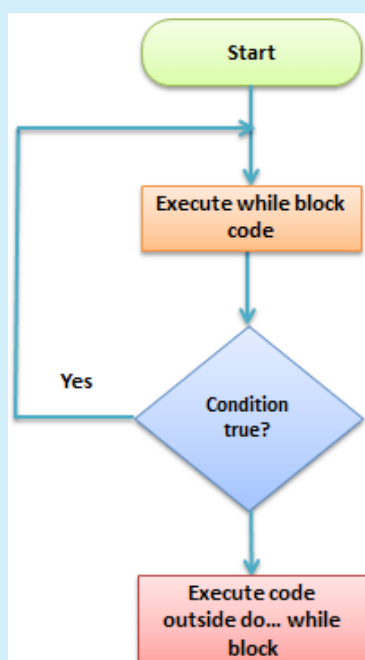
Давайте теперь посмотрим на основной синтаксис цикла do... while

```
<?php  
do{  
block of code to be executed  
} while(condition);  
?>
```

- «Do {...} while (...)» - это код блока цикла do... while
- «condition» - это условие, которое будет оцениваться циклом while
- «block of code ...» - это код, который выполняется хотя бы один раз циклом do... while

## Как это работает

Блок-схема, показанная ниже, показывает, как работает цикл while ...



## Практический пример

Теперь мы собираемся изменить пример цикла `while...` и реализовать его с помощью цикла `do... while` и установить начальное значение счетчика равным 9.

Код ниже реализует приведенный выше модифицированный пример

```
<?php
$i = 9;

do{
    echo "$i is". " <br>";
}while($i < 9);

?>
```

Вышеприведенный код выводит:

9

**Обратите внимание, что** в приведенном выше примере выводятся только 9.

Это связано с тем, что цикл `do... while` выполняется хотя бы один раз, даже если заданное условие оценивается как ложное.

## Резюме

- Цикл `for...` используется для выполнения блока указанное количество раз
- Цикл `foreach...` используется для обхода массивов
- Цикл `while...` используется для выполнения блока кода, если заданное условие считается ложным
- Цикл `do... while` используется для выполнения блока кода хотя бы один раз, тогда как остальная часть выполнения зависит от оценки установленного условия

## Альтернативный синтаксис для структур управления

### Синтаксис

- структура: `/ * код * / endstructure;`

### замечания

При смешивании альтернативной структуры для switch с HTML важно не иметь пробелов между начальным `switch($condition):` и первым `case $value:`. Это делается для того, чтобы повторить что-то (пробелы) перед случаем.

Все структуры управления следуют одной и той же общей идее. Вместо того, чтобы использовать фигурные скобки для инкапсуляции кода, вы используете двоеточие и `endstructure; statement:`

```
structure: /* code */ endstructure;
```

### Examples

#### Альтернатива для for

```
<?php
for ($i = 0; $i < 10; $i++):
    do_something($i);
endfor;
?>
```

```
<?php for ($i = 0; $i < 10; $i++): ?>

<p>Do something in HTML with <?php echo $i; ?></p>

<?php endfor; ?>
```

#### Альтернативный оператор while

```
<?php
while ($condition):
    do_something();
endwhile;
?>
```

```
<?php while ($condition): ?>

<p>Do something in HTML</p>

<?php endwhile; ?>
```

## Альтернативный оператор foreach

```
<?php  
foreach ($collection as $item):  
    do_something($item);  
endforeach;  
?>
```

```
<?php foreach ($collection as $item): ?>  
  
<p>Do something in HTML with <?php echo $item; ?></p>  
  
<?php endforeach; ?>
```

## Альтернативный оператор switch

```
<?php  
switch ($condition):  
    case $value:  
        do_something();  
        break;  
    default:  
        do_something_else();  
        break;  
endswitch;  
?>
```

```
<?php switch ($condition): ?>  
  
<?php case $value: /* having whitespace before your cases will cause an error */ ?>  
  
<p>Do something in HTML</p>  
  
<?php break; ?>  
  
<?php default: ?>  
  
<p>Do something else in HTML</p>  
  
<?php break; ?>  
  
<?php endswitch; ?>
```

## Альтернативный оператор if / else

```
<?php
if ($condition):
    do_something();
elseif ($another_condition):
    do_something_else();
else:
    do_something_different();
endif;
?>
```

```
<?php if ($condition): ?>
    <p>Do something in HTML</p>
<?php elseif ($another_condition): ?>
    <p>Do something else in HTML</p>
https://riptutorial.com/ru/home 109
<?php else: ?>
    <p>Do something different in HTML</p>
<?php endif; ?>
```