



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)
- [Help](#)
- 

[Dutch PHP Conference 2024](#)

[Getting Started](#)

[Introduction](#)

[A simple tutorial](#)

[Language Reference](#)

[Basic syntax](#)

[Types](#)

[Variables](#)

[Constants](#)

[Expressions](#)

[Operators](#)

[Control Structures](#)

[Functions](#)

[Classes and Objects](#)

[Namespaces](#)

[Enumerations](#)

[Errors](#)

[Exceptions](#)

[Fibers](#)

[Generators](#)

[Attributes](#)

[References Explained](#)

[Predefined Variables](#)

[Predefined Exceptions](#)

[Predefined Interfaces and Classes](#)

[Predefined Attributes](#)

[Context options and parameters](#)

[Supported Protocols and Wrappers](#)

[Security](#)

[Introduction](#)

[General considerations](#)

[Installed as CGI binary](#)

[Installed as an Apache module](#)

[Session Security](#)

[Filesystem Security](#)

[Database Security](#)

[Error Reporting](#)

[User Submitted Data](#)

[Hiding PHP](#)

[Keeping Current](#)

[Features](#)

[HTTP authentication with PHP](#)

[Cookies](#)

[Sessions](#)

[Dealing with XForms](#)

[Handling file uploads](#)

[Using remote files](#)

[Connection handling](#)

[Persistent Database Connections](#)

[Command line usage](#)

[Garbage Collection](#)
[DTrace Dynamic Tracing](#)

[Function Reference](#)

[Affecting PHP's Behaviour](#)
[Audio Formats Manipulation](#)
[Authentication Services](#)
[Command Line Specific Extensions](#)
[Compression and Archive Extensions](#)
[Cryptography Extensions](#)
[Database Extensions](#)
[Date and Time Related Extensions](#)
[File System Related Extensions](#)
[Human Language and Character Encoding Support](#)
[Image Processing and Generation](#)
[Mail Related Extensions](#)
[Mathematical Extensions](#)
[Non-Text MIME Output](#)
[Process Control Extensions](#)
[Other Basic Extensions](#)
[Other Services](#)
[Search Engine Extensions](#)
[Server Specific Extensions](#)
[Session Extensions](#)
[Text Processing](#)
[Variable and Type Related Extensions](#)
[Web Services](#)
[Windows Only Extensions](#)
[XML Manipulation](#)
[GUI Extensions](#)

Keyboard Shortcuts

?	This help
j	Next menu item
k	Previous menu item
g p	Previous man page
g n	Next man page
G	Scroll to bottom
g g	Scroll to top
g h	Goto homepage
g s	Goto search (current page)
/	Focus search box

[foreach »](#)
[« do-while](#)

- [Руководство по PHP](#)
- [Справочник языка](#)
- [Управляющие конструкции](#)

Change language: Russian ▾

for

(PHP 4, PHP 5, PHP 7, PHP 8)

Цикл `for` самый сложный цикл в PHP. Он ведёт себя так же, как и в языке C. Синтаксис цикла `for` следующий:

```
for (expr1; expr2; expr3)
    statement
```

Первое выражение (*expr1*) всегда вычисляется (выполняется) только один раз в начале цикла.

В начале каждой итерации оценивается выражение *expr2*. Если оно принимает значение **true**, то цикл продолжается и выполняются вложенные операторы. Если оно принимает значение **false**, выполнение цикла заканчивается.

В конце каждой итерации выражение *expr3* вычисляется (выполняется).

Каждое из выражений может быть пустым или содержать несколько выражений, разделённых запятыми. В *expr2* все выражения, разделённые запятыми, вычисляются, но результат берётся из последнего. Если выражение *expr2* отсутствует, это означает, что цикл будет выполняться бесконечно. (PHP неявно воспринимает это значение как **true**, так же, как в языке C). Это может быть не так бесполезно, как вы могли подумать, так как часто необходимо прервать цикл, используя условный оператор [break](#) вместо использования выражения в цикле `for`, которое принимает истинное значение.

Рассмотрим следующие примеры. Все они отображают числа от 1 до 10:

```
<?php
/* пример 1 */

for ($i = 1; $i <= 10; $i++) {
    echo $i;
}

/* пример 2 */

for ($i = 1; ; $i++) {
    if ($i > 10) {
        break;
    }
    echo $i;
}

/* пример 3 */

$i = 1;
for ( ; ; ) {
    if ($i > 10) {
        break;
    }
    echo $i;
    $i++;
}

/* пример 4 */

for ($i = 1, $j = 0; $i <= 10; $j += $i, print $i, $i++);
?>
```

Конечно, первый пример кажется самым хорошим (или, возможно, четвёртый), но вы можете обнаружить, что возможность использовать пустые выражения в циклах `for` может стать удобной во многих случаях.

PHP также поддерживает альтернативный синтаксис с двоеточием для циклов `for`.

```
for (expr1; expr2; expr3):
    statement
```

```
...
endfor;
```

Перебор массивов как показано ниже - это обычное дело для многих пользователей.

```
<?php
/*
 * Это массив с некоторыми данными, которые мы хотим изменить
 * при работе цикла.
 */
$people = array(
array('name' => 'Kalle', 'salt' => 856412),
array('name' => 'Pierre', 'salt' => 215863)
);

for($i = 0; $i < count($people); ++$i) {
$people[$i]['salt'] = mt_rand(000000, 999999);
}
?>
```

Вышеприведённый код может работать медленно, так как размер массива вычисляется в каждой итерации. Поскольку размер не меняется, цикл может быть легко оптимизирован с помощью промежуточной переменной, в которую будет записан размер массива, вместо повторяющихся вызовов функции [count\(\)](#):

```
<?php
$people = array(
array('name' => 'Kalle', 'salt' => 856412),
array('name' => 'Pierre', 'salt' => 215863)
);

for($i = 0, $size = count($people); $i < $size; ++$i) {
$people[$i]['salt'] = mt_rand(000000, 999999);
}
?>
```

[+add a note](#)

User Contributed Notes 2 notes

[up](#)

[down](#)

311

[matthiaz ¶](#)

12 years ago

Looping through letters is possible. I'm amazed at how few people know that.

```
for($col = 'R'; $col != 'AD'; $col++) {
echo $col.' ';
}
```

returns: R S T U V W X Y Z AA AB AC

Take note that you can't use `$col < 'AD'`. It only works with `!=`.
Very convenient when working with excel columns.

[up](#)

[down](#)

71

[nzamani at cyberworldz dot de ¶](#)

22 years ago

The point about the speed in loops is, that the middle and the last expression are executed EVERY time it loops.
So you should try to take everything that doesn't change out of the loop.
Often you use a function to check the maximum of times it should loop. Like here:

```
<?php
```

```
for ($i = 0; $i <= somewhat_calcMax(); $i++) {  
    somewhat_doSomethingWith($i);  
}  
?>
```

Faster would be:

```
<?php  
$maxI = somewhat_calcMax();  
for ($i = 0; $i <= $maxI; $i++) {  
    somewhat_doSomethingWith($i);  
}  
?>
```

And here a little trick:

```
<?php  
$maxI = somewhat_calcMax();  
for ($i = 0; $i <= $maxI; somewhat_doSomethingWith($i++)) ;  
?>
```

The `$i` gets changed after the copy for the function (post-increment).

[+add a note](#)

- [Управляющие конструкции](#)
 - [Введение](#)
 - [if](#)
 - [else](#)
 - [elseif/else if](#)
 - [Альтернативный синтаксис управляющих структур](#)
 - [while](#)
 - [do-while](#)
 - [for](#)
 - [foreach](#)
 - [break](#)
 - [continue](#)
 - [switch](#)
 - [match](#)
 - [declare](#)
 - [return](#)
 - [require](#)
 - [include](#)
 - [require_once](#)
 - [include_once](#)
 - [goto](#)
- [Copyright © 2001-2024 The PHP Group](#)
- [My PHP.net](#)
- [Contact](#)
- [Other PHP.net sites](#)
- [Privacy policy](#)

