if statement condition optimisation

Asked 16 years, 3 months ago Modified 5 years, 11 months ago Viewed 10k times

Part of PHP Collective

5

I have an if statement with two conditions (separated by an OR operator), one of the conditions covers +70% of situations and takes far less time to process/execute than the second condition, so in the interests of speed I only want the second condition to be processed if the first condition evaluates to false.

if I order the conditions so that the first condition (the quicker one) appears in the if statement first - on the occasions where this condition is met and evaluates true is the second condition even processed?

```
if ( (condition1) | (condition2) ){
  // do this
}
```

or would I need to nest two if statements to only check the second condition if the first evaluates to false?

```
if (condition1){
  // do this
}else if (condition2){
  // do this
}
```

I am working in PHP, however, I assume that this may be language-agnostic.

PHP **php** **language-agnostic** **conditional-statements**

Share

Improve this question

Follow

edited Jan 18, 2019 at 11:04

Ijas Ameenudeen

9,259 ● 3 ● 43 ● 55

asked Aug 29, 2008 at 18:21

JimmyJ

4,441 ● 3 ● 28 ● 25

11 Answers

Sorted by: Highest score (default)

9

For C, C++, C#, Java and other .NET languages boolean expressions are optimised so that as soon as enough is known nothing else is evaluated.

An old trick for doing obfuscated code was to use this to create if statements, such as:

```
a || b();
```

if "a" is true, "b()" would never be evaluated, so we can rewrite it into:

```
if(!a)
    b();
```

and similarly:

```
a && b();
```

would become

```
if(a)
    b();
```

Please note that this is only valid for the || and && operator. The two operators | and & is bitwise or, and and, respectively, and are therefore not "optimised".

EDIT: As mentioned by others, trying to optimise code using short circuit logic is very rarely well spent time.

First go for clarity, both because it is easier to read and understand. Also, if you try to be too clever a simple reordering of the terms could lead to wildly different behaviour without any apparent reason.

Second, go for optimisation, but only after timing and profiling. Way too many developer do premature optimisation without profiling. Most of the time it's completely useless.

Share
Improve this answer
Follow

edited Aug 29, 2008 at 18:52

answered Aug 29, 2008 at 18:27

Mats Fredriksson
20.1k ● 6 ● 38 ● 57

3

Pretty much every language does a short circuit evaluation. Meaning the second condition is only evaluated if it's aboslutely necessary to. For this to work, most languages use the double pipe, ||, not the single one, |.

See http://en.wikipedia.org/wiki/Short-circuit_evaluation

Share Improve this answer Follow

answered Aug 29, 2008 at 18:24

ChrisLively
88k • 27 • 173 • 247

3

In C, C++ and Java, the statement:

```
if (condition1 | condition2) {
  ...
}
```

will evaluate both conditions every time and only be true if the entire expression is true.

The statement:

```
if (condition1 || condition2) {
  ...
}
```

will evaluate `condition2` only if `condition1` is false. The difference is significant if condition2 is a function or another expression with a side-effect.

There is, however, no difference between the `||` case and the `if` / `else` case.

Share Improve this answer Follow

answered Aug 29, 2008 at 18:34

Anders Sandvig
21k • 16 • 61 • 74

2

I've seen a lot of these types of questions lately--optimization to the nth degree.

I think it makes sense in certain circumstances:

1. Computing condition 2 is not a constant time operation
2. You are asking strictly for educational purposes--you want to know how the language works, not to save 3us.

In other cases, worrying about the "fastest" way to iterate or check a conditional is silly. Instead of writing tests which require millions of trials to see any recordable (but insignificant) difference, focus on clarity.

When someone else (could be you!) picks up this code in a month or a year, what's going to be most important is clarity.

In this case, your first example is shorter, clearer and doesn't require you to repeat yourself.

Share Improve this answer Follow

answered Aug 29, 2008 at 18:32

Michael Haren
108k ● 41 ● 171 ● 206

2

According to this article PHP does short circuit evaluation, which means that if the first condition is met the second is not even evaluated. It's quite easy to test also (from the article):

```
<?php
/* ch06ex07 – shows no output because of short circuit evaluation */

if (true || $intVal = 5) // short circuits after true
{

echo $intVal; // will be empty because the assignment never took place
}

?>
```

Share Improve this answer Follow

answered Aug 29, 2008 at 18:32

Farinha
18.1k ● 22 ● 66 ● 80

2

The short-circuiting is not for optimization. It's main purpose is to avoid calling code that will not work, yet result in a readable test. Example:

```
if (i < array.size() && array[i]==foo) ...
```

Note that array[i] may very well get an access violation if i is out of range and crash the program. Thus this program is certainly depending on short-circuiting the evaluation!

I believe this is the reason for writing expressions this way far more often than optimization concerns.

Share Improve this answer Follow

answered Sep 16, 2008 at 21:48

spitzak

1

While using short-circuiting for the purposes of optimization is often overkill, there are certainly other compelling reasons to use it. One such example (in C++) is the following:

```
if( pObj != NULL && *pObj == "username" ) {
    // Do something...
}
```

Here, short-circuiting is being relied upon to ensure that `pObj` has been allocated prior to dereferencing it. This is far more concise than having nested `if` statements.

Share Improve this answer Follow

answered Aug 29, 2008 at 22:00

Matt Dillard

14.9k ● 7 ● 52 ● 61

0

Since this is tagged language agnostic I'll chime in. For Perl at least, the first option is sufficient, I'm not familiar with PHP. It evaluates left to right and drops out as soon as the condition is met.

Share Improve this answer Follow

answered Aug 29, 2008 at 18:22

dagorym

5,795 ● 3 ● 26 ● 23

0

In most languages with decent optimization the former will work just fine.

Share Improve this answer Follow

answered Aug 29, 2008 at 18:26

levand

8,480 ● 3 ● 42 ● 55

0

The `|` is a bitwise operator in PHP. It does not mean `$a OR $b`, exactly. You'll want to use the double-pipe. And yes, as mentioned, PHP does short-circuit evaluation. In similar fashion, if the first condition of an `&&` clause evaluates to false, PHP does not evaluate the rest of the clause, either.

Share Improve this answer Follow

answered Aug 29, 2008 at 18:45

Brian Warshaw

23k ● 9 ● 54 ● 72

VB.net has two wonderful expression called "OrElse" and "AndAlso"

OrElse will short circuit itself the first time it reaches a True evaluation and execute the code you desire.

```
If FirstName = "Luke" OrElse FirstName = "Darth" Then
   Console.Writeline "Greetings Exalted One!"
End If
```

AndAlso will short circuit itself the first time it a False evaluation and not evaluate the code within the block.

```
If FirstName = "Luke" AndAlso LastName = "Skywalker" Then
   Console.Writeline "You are the one and only."
End If
```

I find both of these helpful.

Share Improve this answer Follow

answered Aug 29, 2008 at 22:06

Dillie-O

29.7k ● 14 ● 102 ● 141

1 That's exactly what || and && do in most other languages, respectively. – Arda Xi Jun 12, 2010 at 9:49