Should a developer aim for readability or performance first? [closed]

Asked 16 years, 2 months ago Modified 10 years, 6 months ago

Viewed 19k times

104

As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, visit the help center for guidance.

Closed 11 years ago.

Oftentimes a developer will be faced with a choice between two possible ways to solve a problem -- one that is idiomatic and readable, and another that is less intuitive, but may perform better. For example, in C-based languages, there are two ways to multiply a number by 2:

```
int SimpleMultiplyBy2(int x)
{
    return x * 2;
}
```

and

```
int FastMultiplyBy2(int x)
{
    return x << 1;
}
```

The first version is simpler to pick up for both technical and non-technical readers, but the second one may perform better, since bit shifting is a simpler operation than multiplication. (For now, let's assume that the compiler's optimizer would not detect this and optimize it, though that is also a consideration).

As a developer, which would be better as an initial attempt?

performance **readability**

Share

Improve this question

Follow

edited Oct 8, 2008 at 16:11

community wiki
3 revs
JohnMcG

A bit harsh. Good question about a concern we all have at times. +1 – Inisheer Oct 8, 2008 at 15:01

3 This example is obviously contrived and trivial. You wouldn't really have a function with a hard-coded multiplier. – JohnMcG Oct 8, 2008 at 15:11

1 The point is I see a lot of questions like "does < perform better than <=?" This is the wrong question -- the right (first) question is which is idiomatic or conventional, then worry about performance. – JohnMcG Oct 8, 2008 at 15:14

2 This is one of the best questions that I have read on stackoverflow. This gets to the meat of how computers operate, not just on the semantics of the language. +1 – WolfmanDragon Oct 8, 2008 at 15:25

2 @OutlawLemur I'm aware of that. But some people ask whether it would be better to, for example, construct loops using < or <= (with the comparison value being incremented beforehand in the latter case). – JohnMcG May 21, 2012 at 14:39

35 Answers

Sorted by: Highest score (default)

132

You missed one.

First code for correctness, then for clarity (the two are often connected, of course!). Finally, and only if you have real empirical evidence that you actually need to, you can look at optimizing. Premature optimization really is evil. Optimization almost always costs you time, clarity, maintainability. You'd better be sure you're buying something worthwhile with that.

Note that good algorithms almost always beat localized tuning. There is no reason you can't have code that is correct, clear, and fast. You'll be unreasonably lucky to get there starting off focusing on `fast' though.

Share Improve this answer Follow

edited Oct 9, 2008 at 15:13

community wiki
2 revs
simon

This is far and away the best answer here. Tweak the algorithm, not the code. With subtle changes I can make a jscript Sieve of Erastosthenes outperform an otherwise identical C++ version. (Not the Sieve of Atkins, my own method.) – Peter Wone Oct 9, 2008 at 9:13

2 Too bad you can't favorite responses. :) – Sandor Davidhazi Oct 13, 2008 at 9:12

Incorrect. First code for readablility. Readable is maintainable. If the code is a jumbled mess of spaghetti, it's a pain to do anything. correctness second, optimization last. – ThaJay Feb 13, 2023 at 10:20

really now I get why they say 'optimize what actually gets called and take cycles' or in other words use the profiler – CrazyTard1990 Jul 7 at 15:34

IMO the obvious readable version first, until performance is measured and a faster version is required.

64

Share Improve this answer

Follow

answered Oct 8, 2008 at 14:56

community wiki
kenny

> I agree. Last year I implemented a major component as apart of my companies server-side Java code-base and did my best to make it readible. It later turned out there were performance issues and did some major reworking of its design that lead to something that is a bit less readible in some ways. – Ryan Delucchi Oct 8, 2008 at 20:12

46

Take it from Don Knuth

> Premature optimization is the root of all evil (or at least most of it) in programming.

Share Improve this answer

Follow

answered Oct 8, 2008 at 14:59

community wiki
Ryan

The quote is not from Don itself, but from Hoare. Don just made it popular. Check wikipedia. – kohlerm Oct 8, 2008 at 19:27

1 And it's a selective quotation of one *clause* from an entire paragraph, which contains some very important qualifications. – user207421 Jan 31, 2016 at 23:55

25

Readability 100%

If your compiler can't do the "x*2" => "x <<1" optimization for you -- get a new compiler!

Also remember that 99.9% of your program's time is spent waiting for user input, waiting for database queries and waiting for network responses. Unless you are doing the multiple 20 bajillion times, it's not going to be noticeable.

Share Improve this answer Follow

edited Jun 7, 2014 at 4:48

community wiki
2 revs, 2 users 91%
James Curran

13

Readability for sure. Don't worry about the speed unless someone complains

Share Improve this answer

answered Oct 8, 2008 at 15:00

Follow

community wiki
Miles

10

In your given example, 99.9999% of the compilers out there will generate the same code for both cases. Which illustrates my general rule - write for readability and maintainability first, and optimize only when you need to.

Share Improve this answer

Follow

answered Oct 8, 2008 at 14:58

community wiki
Paul Tomblin

C compilers will compile into a different assembly code for the two examples shown. The first creates a loop, while the second creates a shift left instruction. This should be true for All C style compilers, as I have not tested each one, I can make no promises on that. – WolfmanDragon Oct 8, 2008 at 15:33

For this specific example, certainly. There are lots of cases where that isn't, so the general question is still a good one – Mark Baker Oct 8, 2008 at 15:33

@WolfmanDragon, what the hell are you talking about? Why would "* 2" generate a loop? When I try it with "gcc -O2 -s", I get addl instructions in both cases. – Paul Tomblin Oct 8, 2008 at 16:44

1 If your compiler creates a loop in that function, I'd recommend you get another compiler! – Martin Vilcans Feb 21, 2009 at 20:41

9

Readability.

Coding for performance has it's own set of challenges. Joseph M. Newcomer said it well

> Optimization matters only when it matters. When it matters, it matters a lot, but until you know that it matters, don't waste a lot of time doing it. Even if you know it matters, you need to know where it matters. Without performance data, you won't know what to optimize, and you'll probably optimize the wrong thing.
>
> The result will be obscure, hard to write, hard to debug, and hard to maintain code that doesn't solve your problem. Thus it has the dual disadvantage of (a) increasing software development and software maintenance costs, and (b) having no performance effect at all.

Share Improve this answer Follow

answered Oct 8, 2008 at 17:12

community wiki
nwahmaet

I would go for **readability** first. Considering the fact that with the kind of optimized languages and hugely loaded

7

machines we have in these days, most of the code we write in readable way will perform decently.

In some very rare scenarios, where you are pretty sure you are going to have some performance bottle neck (may be from some past bad experiences), and you managed to find some weird trick which can give you huge performance advantage, you can go for that. But you should comment that code snippet very well, which will help to make it more readable.

Share Improve this answer

Follow

edited Oct 8, 2008 at 18:06

community wiki
3 revs
Vijesh VP

6

Readability. The time to optimize is when you get to beta testing. Otherwise you never really know what you need to spend the time on.

Share Improve this answer

Follow

answered Oct 8, 2008 at 14:57

community wiki
ChrisLively

4

A often overlooked factor in this debate is the extra time it takes for a programmer to navigate, understand and modify less readible code. Considering a programmer's time goes for a hundred dollars an hour or more, this is a very real cost.
Any performance gain is countered by this direct extra cost in development.

Share Improve this answer Follow

answered Oct 8, 2008 at 15:00

community wiki
Rik

4

Putting a comment there with an explanation would make it readable and fast.

It really depends on the type of project, and how important performance is. If you're building a 3D game, then there are usually a lot of common optimizations that you'll want to throw in there along the way, and there's no reason not to (just don't get too carried away early). But if you're doing something tricky, comment it so anybody looking at it will know how and why you're being tricky.

Share Improve this answer Follow

answered Oct 8, 2008 at 15:09

community wiki

Gerald

4

Both. Your code should balance both; readability and performance. Because ignoring either one will screw the ROI of the project, which in the end of the day is all that matters to your boss.

Bad readability results in decreased maintainability, which results in more resources spent on maintenance, which results in a lower ROI.

Bad performance results in decreased investment and client base, which results in a lower ROI.

Share Improve this answer Follow

edited Oct 8, 2008 at 15:38

community wiki
2 revs
Kon

3

The answer depends on the context. In device driver programming or game development for example, the second form is an acceptable idiom. In business applications, not so much.

Your best bet is to look around the code (or in similar *successful* applications) to check how other developers do it.

Share Improve this answer Follow

answered Oct 8, 2008 at 15:19

community wiki
ilitirit

3

If you're worried about readability of your code, don't hesitate to add a comment to remind yourself what and why you're doing this.

Share Improve this answer Follow

answered Oct 8, 2008 at 15:30

community wiki
Michael McCarty

> Indeed, but to clarify: Only comment why. If you're commenting how, the code needs refactoring and names need improving. – ThaJay Feb 13, 2023 at 10:25

3

using << would by a micro optimization. So Hoare's (not Knuts) rule:

Premature optimization is the root of all evil.

applies and you should just use the more readable version in the first place.

This is rule is IMHO often misused as an excuse to design software that can never scale, or perform well.

Share Improve this answer Follow

answered Oct 8, 2008 at 15:33

community wiki
kohlerm

3

Readability is the FIRST target.

In the 1970's the army tested some of the then "new" techniques of software development (top down design, structured programming, chief programmer teams, to name a few) to determine which of these made a statistically significant difference.

THe ONLY technique that made a statistically significant difference in development was...

ADDING BLANK LINES to program code.

The improvement in readability in those pre-structured, pre-object oriented code was the only technique in these studies that improved productivity.

==========

Optimization should only be addressed when the entire project is unit tested and ready for instrumentation. You never know WHERE you need to optimize the code.

In their landmark books Kernigan and Plauger in the late 1970's SOFTWARE TOOLS (1976) and SOFTWARE

TOOLS IN PASCAL (1981) showed ways to create structured programs using top down design. They created text processing programs: editors, search tools, code pre-processors.

When the completed text formating function was INSTRUMENTED they discovered that most of the processing time was spent in three routines that performed text input and output (In the original book, the i-o functions took 89% of the time. In the pascal book, these functions consumed 55%!)

They were able to optimize these THREE routines and produced the results of increased performance with reasonable, manageable development time and cost.

Share Improve this answer Follow

answered Oct 24, 2008 at 19:58

community wiki
SystemSmith

2

The larger the codebase, the more readability is crucial. Trying to understand some tiny function isn't so bad. (Especially since the Method Name in the example gives you a clue.) Not so great for some epic piece of uber code written by the loner genius who just quit coding because he has finally seen the top of his ability's complexity and it's what he just wrote for you and you'll never ever understand it.

Share Improve this answer Follow

answered Oct 8, 2008 at 14:59

community wiki
mspmsp

2

As almost everyone said in their answers, I favor **readability**. 99 out of 100 projects I run have no hard response time requirements, so it's an easy choice.

Before you even start coding you should already know the answer. Some projects have certain performance requirements, like 'need to be able to run task X in Y (milli)seconds'. If that's the case, you have a goal to work towards and you know when you have to optimize or not. (hopefully) this is determined at the requirements stage of your project, not when writing the code.

Good readability and the ability to optimize later on are a result of proper software design. If your software is of sound design, you should be able to isolate parts of your software and rewrite them if needed, without breaking other parts of the system. Besides, most true optimization cases I've encountered (ignoring some real low level tricks, those are incidental) have been in changing from one algorithm to another, or caching data to memory instead of disk/network.

Share Improve this answer Follow

answered Oct 13, 2008 at 8:55

community wiki
Erik van Brakel

2

If there is no readability , it will be very hard to get performance improvement when you really need it.

Performance should be only improved when it is a problem in your program, there are many places would be a bottle neck rather than this syntax. Say you are squishing 1ns improvement on a << but ignored that 10 mins IO time.

Also, regarding readability, a professional programmer should be able to read/understand computer science terms. For example we can name a method enqueue rather than we have to say putThisJobInWorkQueue.

Share Improve this answer Follow

answered Jan 18, 2011 at 8:12

community wiki
Yuan

While I agree, I think you have a poor example. For instance, as someone who knows nothing about your codebase, enqueue is meaning less to me. I don't wanna know how, I wanna know what. The example you gave says the what much better. – Shaun Sweet Oct 19, 2017 at 3:09

1

The bitshift versus the multiplication is a **trivial optimization that gains** next to **nothing**. And, as has been pointed out, your compiler should do that for you. Other than that, the gain is neglectable anyhow as is the CPU this instruction runs on.

On the other hand, if you need to perform serious computation, you will require the right data structures. But if your problem is complex, finding out about that is part of the solution. **As an illustration, consider searching for an ID number in an array of 1000000 unsorted objects. Then reconsider using a binary tree or a hash map.**

But optimizations like n << C are usually neglectible and trivial to change to at any point. Making code readable is not.

Share Improve this answer Follow

answered Oct 8, 2008 at 15:24

community wiki
mstrobl

1

It depends on the task needed to be solved. Usually readability is more importrant, but there are still some tasks when you shoul think of performance in the first place. And you can't just spend a day or to for profiling and optimization after everything works perfectly, because optimization itself may require rewriting sufficiant

part of a code from scratch. But it is not common nowadays.

Share Improve this answer

Follow

answered Oct 8, 2008 at 15:26

community wiki
akalenuk

1

I'd say go for readability.

But in the given example, I think that the second version is already readable enough, since the name of the function exactly states, what is going on in the function.

If we just always had functions that told us, what they do ...

Share Improve this answer

Follow

answered Oct 8, 2008 at 15:35

community wiki
Dan Soap

1

You should always maximally optimize, performance always counts. The reason we have bloatware today, is that most programmers don't want to do the work of optimization.

Having said that, you can always put comments in where slick coding needs clarification.

Share Improve this answer Follow

answered Oct 8, 2008 at 16:34

community wiki
Lance Roberts

I agree on a certain level. I don't think you should put in micro optimizations like the original question stated. You do have to design your system in such a way that you use resources in an optimal way. There's much more performance to gain in that field. – Erik van Brakel Oct 13, 2008 at 8:58

1 We do have bloatware today, but I don't blame it on lack of optimization. I blame it on overdesign, swatting flies with bazookas, building ocean liners when only rowboats are needed. Then of course it's ponderous. – Mike Dunlavey Dec 9, 2008 at 20:32

I would agree with you both in that optimization of design is the first priority. I would also say that that same attitude should be applied to all levels of the software engineering process. If you don't bother to do optimization at the code level, then you're probably lacking during design also. – Lance Roberts Dec 10, 2008 at 16:32

1

There is no point in optimizing if you don't know your bottlenecks. You may have made a function incredible efficient (usually at the expense of readability to some degree) only to find that portion of code hardly ever runs, or it's spending more time hitting the disk or database than you'll ever save twiddling bits. So you can't micro-optimize until you have something to measure, and then you might as well start off for readability. However, you should be mindful of both speed and understandability when designing the overall architecture, as both can have a massive impact and be difficult to change (depending on coding style and methedologies).

Share Improve this answer Follow

answered Oct 8, 2008 at 22:01

community wiki
ICR

1

It is estimated that about 70% of the cost of software is in maintenance. Readability makes a system easier to maintain and therefore brings down cost of the software over its life.

There are cases where performance is more important the readability, that said they are few and far between.

Before sacrifing readability, think "Am I (or your company) prepared to deal with the extra cost I am adding to the system by doing this?"

Share Improve this answer

Follow

answered Oct 9, 2008 at 3:35

community wiki
Peter Tomlins

1

I don't work at google so I'd go for the evil option. (optimization)

In Chapter 6 of Jon Bentley's "Programming Pearls", he describes how one system had a 400 times speed up by optimizing at 6 different design levels. I believe, that by not caring about performance at these 6 design levels, modern implementors can easily achieve 2-3 orders of magnitude of slow down in their programs.

Share Improve this answer

Follow

answered Oct 13, 2008 at 8:30

community wiki
paperhorse

1

Readability first. But even more than readability is simplicity, *especially* in terms of data structure.

I'm reminded of a student doing a vision analysis program, who couldn't understand why it was so slow. He merely followed good programming practice - each pixel

was an object, and it worked by sending messages to its neighbors...

check this out

Share Improve this answer Follow

answered Nov 4, 2008 at 22:48

community wiki
Mike Dunlavey

0

Write for readability first, but expect the readers to be *programmers*. Any programmer worth his or her salt should know the difference between a multiply and a bitshift, or be able to read the ternary operator where it is used appropriately, be able to look up and understand a complex algorithm (you are commenting your code right?), etc.

Early over-optimization is, of course, quite bad at getting you into trouble later on when you need to refactor, but that doesn't really apply to the optimization of individual methods, code blocks, or statements.

Share Improve this answer Follow

answered Oct 8, 2008 at 15:11

community wiki
wprl

How much does an hour of processor time cost?

How much does an hour of programmer time cost?

Share Improve this answer Follow

answered Oct 8, 2008 at 15:43

community wiki
Andy Lester

1 how much does an hour of end-user time cost? now mutiply that by the number of users. – gbjbaanb Oct 8, 2008 at 16:39

gbjbaanb: My thoughts exactly. Andy's comment only works for services the end user will never see, and even then it's hardly a good comparison. – Erik van Brakel Oct 13, 2008 at 8:47

0

IMHO both things have nothing to do. You should first go for code that works, as this is more important than performance or how well it reads. Regarding readability: your code should always be readable in any case.

However I fail to see why code can't be readable and offer good performance at the same time. In your example, the second version is as readable as the first one to me. What is less readable about it? If a programmer doesn't know that shifting left is the same as multiplying by a power of two and shifting right is the same as dividing by a power of two... well, then you have much more basic problems than general readability.

Share Improve this answer

Follow

answered Oct 8, 2008 at 16:16

community wiki
Mecki