[0060] Premature Optimization: The Root of All Evil

Donald Knuth - a famous computer scientist in the 1970s - once said: "Premature optimization is the root of all evil." His words were very true then and they are still true today.

In programming, almost every line of code you write involves some kind of trade-off. Over time, good programmers get better at recognizing and understanding the trade-offs that they make when they write code.

Kinds of Optimization

Experienced programmers recognize that there are many different "dimensions" to the code they write and, as they are creating thier code, they are making choices that favor one dimension over the others. The optimization choices can have a profound effect on how the program works. Examples of some of the "dimensions" that programs can be optimized for include:

- Speed
- Size
- Time to Market
- Portability to other run-time environments
- Readability
- Maintainability
- Modifyability
- Cleverness
- Ease of Programming
- Minimal cost
- Small runtime memory footprint
- Compatibility with specific run-time environments
- Minimal saved state
- Reduced start-up time
- Reduced shut-down time
- Provable Correctness
- and more...

Often people optimize code for some combination of several dimensions. The trick is to know which dimensions matter and which don't.

Professor Knuth's observation concerns a common trap that new programmers get into where they assume that they know exactly what to optimize for (typically speed). When faced with a choice about how their program can complete a task, they will immediately try to write code that runs quickly - often forgetting to consider the other optimization dimensions.

What's Most Important to Many Businesses These Days

Nowadays, many computers have lots of memory, lots of disk space, and lots of idle, unused CPU cycles. That wasn't always the case - the original Apple][had 32K of RAM memory, a very slow 8-bit CPU, and a cassette tape drive for storage. In those days, programs had to be small and fast. There wasn't any other choice. But again, today's computers do not have any of those limitations.

What hasn't changed is the importance of updating programs with bug-fixes, security patches, and new features over time. These changes could be part of a maintenance release, an emergency patch, a new version or a completely new program. Regardless, nowadays, ***the need to quickly understand and update a program is frequently the most important aspect of a program's implementation.***

Note: This statement isn't always true for all companies or all situations. Programmers need to fully understand the environment that their program will run in in order to ensure they are making the right trade-offs.

For this class, we will emphasize writing code that is easy to understand and easy to modify - i.e., we will optimize for Readability and Maintainability. Specifically, within reasonable limits, we will not worry about a program's execution speed.