“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

– Martin Fowler
“Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live.

Code for readability.”

(Author unclear)
Coding Style: Understandability

Anti-Patterns:
- “Style doesn’t matter; it passes all the tests”
- Code that is clever instead of clear

“There are two ways of constructing a software design: one way is to make it so simple that there are **obviously no deficiencies** and the other way is to make it so complicated that there are **no obvious deficiencies**.”

— C.A.R. (Tony) Hoare, 1980 Turing Award Talk

Other people must understand your code
- Peer reviews won’t work if nobody can read your code
  - Write code so that others can tell it is obviously correct
- If others can’t understand it, they will inject bugs
- If you have to think about whether it’s right, then it’s wrong
Make Code Easy To Read

- **Consistent formatting**
  - Consistent indentation, braces
  - Templated headers for files and functions
  - Spaces and "()" to avoid precedence confusion
  - Use space instead of tab

- **Comments**
  - Explain what & why, not just code paraphrase
  - Comments are not a design

- **Naming**
  - Descriptive, consistent naming conventions
    - E.g., variables are nouns; functions are verbs

- **Avoid magic numbers (use const)**
  - Avoid macros (use inline)
Good Code Hygiene

- Modularity
  - Many smaller .c/.cpp files (one per class)
  - Externally visible declarations into .h file

- Conditional Statements
  - Boolean conditional expression results; no assignments
  - All switch statements have a default (usually error trap)
  - Limited nesting (see also cyclomatic complexity)

- Variables
  - Descriptive names that differ significantly
  - Smallest practicable scope for variables; initialize at point of definition
  - Use typedefs to define narrow types (also use uint32_t, use enum, etc.)
  - Range checks & bounds checks (e.g., buffer overflow)

- Handle errors returned by called functions
"We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%"


**Don’t optimize unless you have performance data**
- Most code doesn’t matter for speed
- Use little or no assembly language. Get a better compiler.

**Optimization makes it hard to know your code is right**
- Do you want correct code or tricky code?
  - (Pick one. Which one is safer?)
- Buy a bigger CPU if you have to

https://xkcd.com/1691/
Coding Understandability Best Practices

- Pick a coding style and follow it
  - Use tool support for language formatting
  - Evaluate naming as part of peer review
  - Comments are there to explain implementation

- The point of good style is to avoid bugs
  - Make it hard for a reviewer to miss a problem
    - Even better, make it easy for a tool to find problems
  - Also need to have a good technical style

- Coding style pitfalls:
  - Optimizing for the author instead of the reviewer
  - Making it too easy to deviate from style rules

Great style depends upon point of view.
KEEP IN MIND THAT I'M SELF-TAUGHT, SO MY CODE MAY BE A LITTLE MESSY.

LEMMIE SEE—I'M SURE IT'S FINE.

...Wow.

This is like being in a house built by a child using nothing but a hatchet and a picture of a house.

IT'S LIKE A SALAD RECIPE WRITTEN BY A CORPORATE LAWYER USING A PHONE AUTOCORRECT THAT ONLY KNEW EXCEL FORMULAS.

IT'S LIKE SOMEONE TOOK A TRANSCRIPT OF A COUPLE ARGUING AT IKEA AND MADE RANDOM EDITS UNTIL IT COMPILED WITHOUT ERRORS.

OKAY, I'LL READ A STYLE GUIDE.

https://xkcd.com/1513/