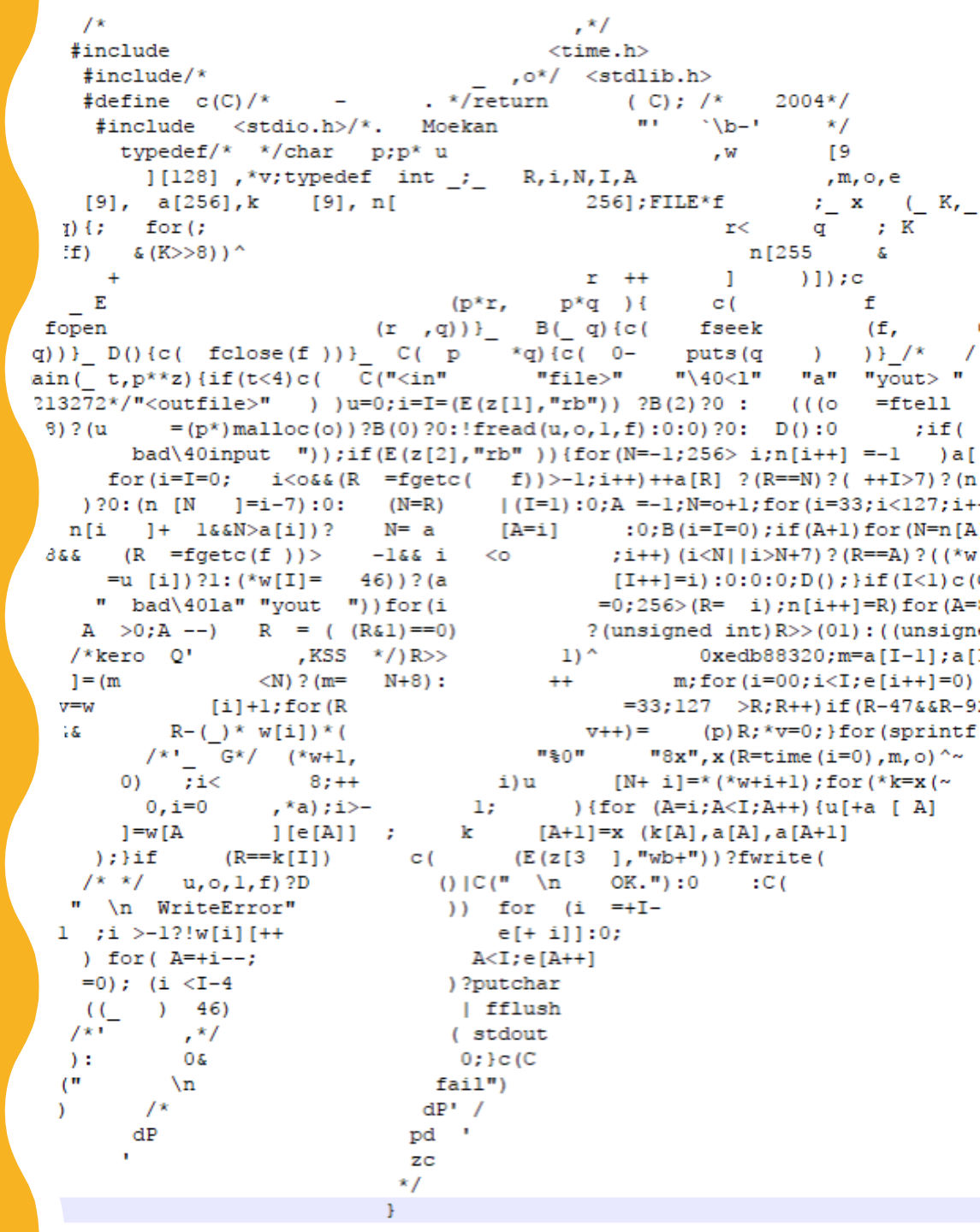
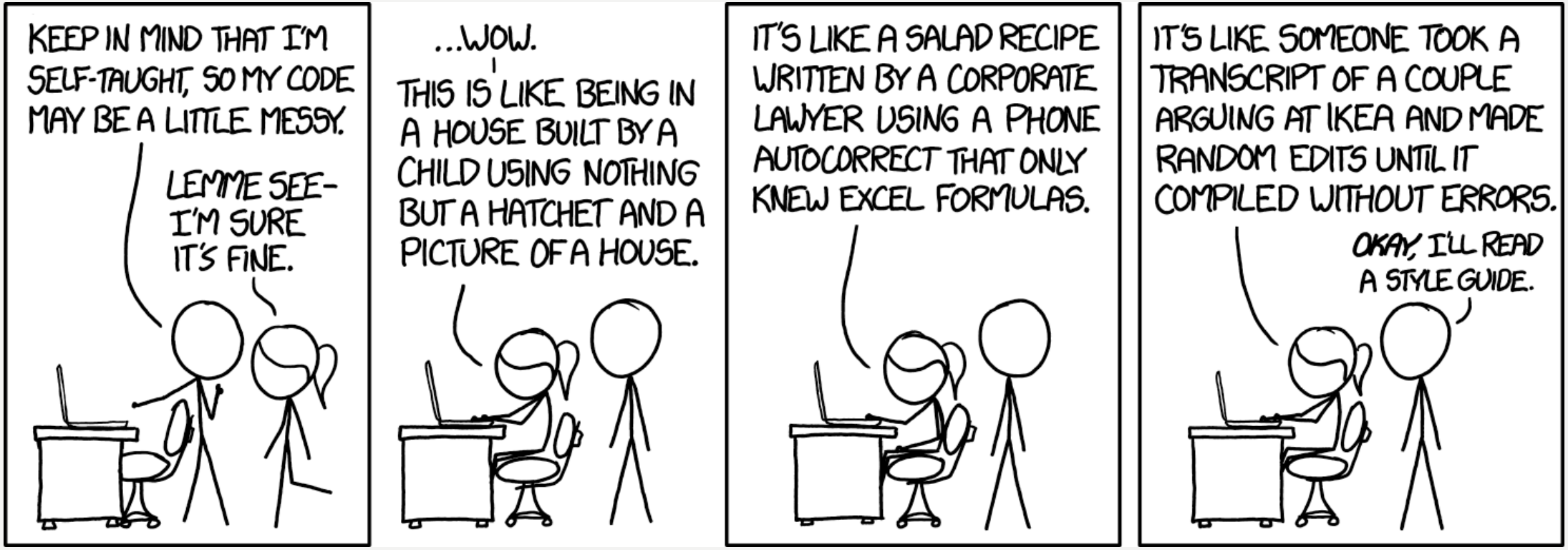
CODE STYLE

ADAM SWEENEY
CS 211
WICHITA STATE UNIVERSITY, EECS



INTRODUCTION (RELEVANT XKCD)



WHAT IS STYLE?

- How the code looks
- Code is hard to read
  - That of other people
  - Your own code a few months later
    - Your own code a couple hours later
- It is important to write code that can be understood
- There are some generally universal standards
  - Usually a few valid options to choose from

WHY STYLE MATTERS

- "Any fool can write code that a computer can understand. Good programmers write code that humans can understand."
  - Martin Fowler, Refactoring: Improving the Design of Existing Code
- Other people will see our code
  - We'll see our code later
- It's extremely important that we write human-readable code

BASICS

INDENTATION

- Choose a size and stick with it
  - Acceptable sizes are 2, 4, or 8 columns
  - Recommend 4
- Exclusively use one of tabs or spaces
  - Many editors have settings to insert spaces or tabs by pressing the tab
  - DO NOT MIX AND MATCH TABS AND SPACES
- Indent only according to your "block-level"

NAMING

- Variables
  - Variables should be given a descriptive, succinct name, usually with an object word
  - Do NOT use Hungarian notation
- Functions
  - Function names should be given a descriptive, succinct name, usually with an action word
    - Some of this changes with Object-Oriented Programming, but we're not doing that

BUT DESCRIPTIVE NAMES ARE LONG

- Not always
  - `tmp` gets its point across just fine
- But there will be times where a variable or function name requires more than one word
- There are choices here
  - Lower camel case [lowerCamelCase] (my recommendation for variables)
  - Upper camel case [UpperCamelCase] (my OOP recommendation for types)
  - Lower snake case [my_function] (my recommendation for functions)
  - Upper snake case [My_Function]

CONSTANTS

- Sometimes we need to work with a number that doesn't change
- We can "name" the number by assigning it to a variable
- We also make the variable `const`, which means it cannot be altered
  - Constant
- `const int NUM_BRANCHES = 10;`
  - All caps is a style recommendation to make constants easier to distinguish from regular variables

BRACE PLACEMENT

- There are many ways to do proper brace placement
  - This class will only consider the two presented as acceptable

```
if (expression) {
    // do stuff
}
```

```
if (expression)
{
    // do stuff
}
```

STILL WITH THE BRACE PLACEMENT

- If you choose to place an open brace on its own line, nothing else goes on that line
- The same goes for the closing brace, with a couple exceptions
  - if/else if blocks
  - do/while loops

```
if (expression) {
    // do stuff
} else if (other expression) {
    // do stuff
} else {
    // do stuff
}
```

```
do {
    // do stuff
}  while (expression is true);
```

TYPES OF COMMENTS

- Single line

```
statement;   // Comment

// Comment
statement;
```

- Multiline

```
/*
 * Beginning of comment
 * Comment body
 * Last line of comment
 */
```

WRITING GOOD COMMENTS

- Proper comments aids in the understanding of code
- Overly verbose/frequent comments or spartan comments hurt readability
- Comments should rarely discuss “what”, but should discuss “how” or “why” instead
  - ”What” is generally plain to see when looking at the code

OTHER TIPS

LOOPS

- By default, your loop counter variable is `i`
  - When dealing with loops, you may choose something more descriptive, but keep it short
  - Ex. Nested loops for rows and columns

```
for (int row = 0; row < …) {
    for (int col = 0; col < …) {
            // do stuff
    }
}
```

BIG IF/ELSE BLOCKS

- Sometimes in big if/else blocks, each decision only needs to do one thing
- Don't have to use braces if you don't need them
  - I recommend always using braces
    - If it's good enough for John Carmack, it's good enough for us
- But if you need them in one branch, you need them in every branch
  - This is a block-by-block decision

BIG IF/ELSE EXAMPLE

```
if (check1)
	// set var
else if (check2)
	// set var
else if (check3) {
	// set var
	// do more stuff
} else
	// set var
```

```
if (check1) {
	// set var
} else if (check2) {
	// set var
} else if (check3) {
	// set var
	// do more stuff
} else {
	// set var
}
```

LOGICAL GROUPS

- This one is harder to quantify
- Programs contain algorithms
  - An algorithm is a series of steps performed to complete a task
- Each step can require multiple lines of code
- Create some white space (one blank line, typically) between logical groups
- When we discuss functions, one blank line between function implementations is an example

CONSISTENCY

- The most important thing in code style is consistency
- Where you have options, make a choice and stick with it
  - I encourage trying different styles, but make your choices on a per assignment basis

REMINDER (RELEVANT XKCD)