Prof. Philip Koopman

Carnegie Mellon University

Electrical & Computer ENGINEERING

Code Style for Humans

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

– Martin Fowler

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 it's not obviously correct, 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)

Obfuscated C Winner: Flight Simulator

```
#include                                    <math.h>
#include                                    <sys/time.h>
#include                                    <X11/Xlib.h>
#include                                    <X11/keysym.h>
                                            double L ,o ,P
                                           ,_=dt,T,Z,D=1,d,
                                           s[999],E,h= 8,I,
                                           J,K,w[999],M,m,O
                                           ,n[999],j=33e-3,i=
                                           1E3,r,t, u,v ,W,S=
                                           74.5,l=221,X=7.26,
                                           a,B,A=32.2,c, F,H;
                                           int N,q, C, y,p,U;
                                          Window z; char f[52]
                                        ; GC k; main(){ Display*e=
 XOpenDisplay( 0); z=RootWindow(e,0); for (XSetForeground(e,k=XCreateGC (e,z,0,0),BlackPixel(e,0))
 ; scanf("%lf%lf%lf",y +n,w+y, y+s)+1; y ++); XSelectInput(e,z= XCreateSimpleWindow(e,z,0,0,400,400,
 0,0,WhitePixel(e,0) ),KeyPressMask); for(XMapWindow(e,z); ; T=sin(O)){ struct timeval G={ 0,dt*1e6}
 ; K= cos(j); N=1e4; M+= H*_; Z=D*K; F+=_*P; r=E*K; W=cos( O); m=K*W; H=K*T; O+=D*_*F/ K+d/K*E*_; B=
 sin(j); a=B*T*D-E*W; XClearWindow(e,z); t=T*E+ D*B*W; j+=d*_*D-_*F*E; P=W*E*B-T*D; for (o+=(I=D*W+E
 *T*B,E*d/K *B+v+B/K*F*D)*_; p<y; ){ T=p[s]+i; E=c-p[w]; D=n[p]-L; K=D*m-B*T-H*E; if(p [n]+w[ p]+p[s
 ]== 0|K <fabs(W=T*r-I*E +D*P) |fabs(D=t *D+Z *T-a *E)> K)N=1e4; else{ q=W/K *4E2+2e2; C= 2E2+4e2/ K
  *D; N-1E4&& XDrawLine(e ,z,k,N ,U,q,C); N=q; U=C; } ++p; } L+=_* (X*t +P*M+m*l); T=X*X+ l*l+M *M;
   XDrawString(e,z,k ,20,380,f,17); D=v/l*15; i+=(B *l-M*r -X*Z)*_; for(; XPending(e); u *=CS!=N){
                                          XEvent z; XNextEvent(e ,&z);
                                              ++*((N=XLookupKeysym
                                                (&z.xkey,0))-IT?
                                                N-LT? UP-N?& E:&
                                                J:& u: &h); --*(
                                                DN -N? N-DT ?N==
                                                RT?&u: & W:&h:&J
                                                ); } m=15*F/l;
                                                c+=(I=M/ l,l*H
                                                +I*M+a*X)*_; H
                                                =A*r+v*X-F*l+(
                                                E=.1+X*4.9/l,t
                                                =T*m/32-I*T/24
                                                 )/S; K=F*M+(
                                                 h* 1e4/l-(T+
                                                 E*5*T*E)/3e2
                                                 )/S-X*d-B*A;
                                                 a=2.63 /l*d;
                                                 X+=( d*l-T/S
                                                  *(.19*E +a
                                                  *.64+J/1e3
                                                  )-M* v +A*
                                                  Z)*_; l +=
                                                  K *_; W=d;
                                                  sprintf(f,
                                                  "%5d  %3d"
                                                  "%7d",p =l
                                                  /1.7,(C=9E3+
                                  O*57.3)%0550,(int)i); d+=T*(.45-14/l*
                                 X-a*130-J* .14)*_/125e2+F*_*v; P=(T*(47
                                 *I-m* 52+E*94 *D-t*.38+u*.21*E) /1e2+W*
                                 179*v)/2312; select(p=0,0,0,0,&G); v-=(
                                  W*F-T*(.63*m-I*.086+m*E*19-D*25-.11*u
                                   )/107e2)*_; D=cos(o); E=sin(o); } }
```

http://blog.aerojockey.com/post/iocccsim

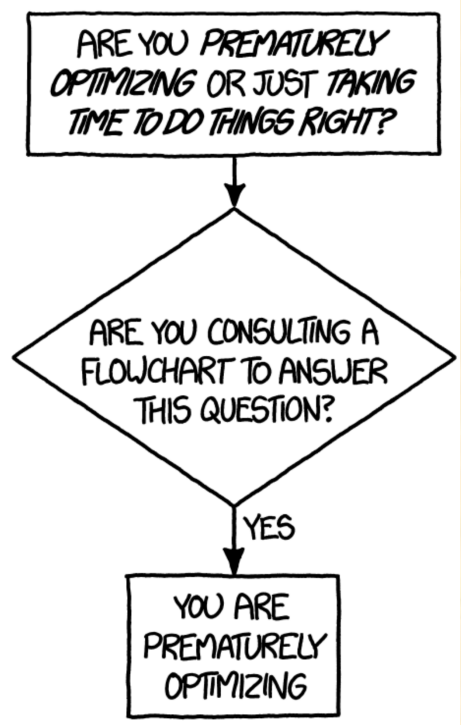
Good Code Hygiene

- 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

Optimization

"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%"

- Donald Knuth (December 1974). "Structured Programming with go to Statements". ACM Journal Computing Surveys 6 (4): 268.



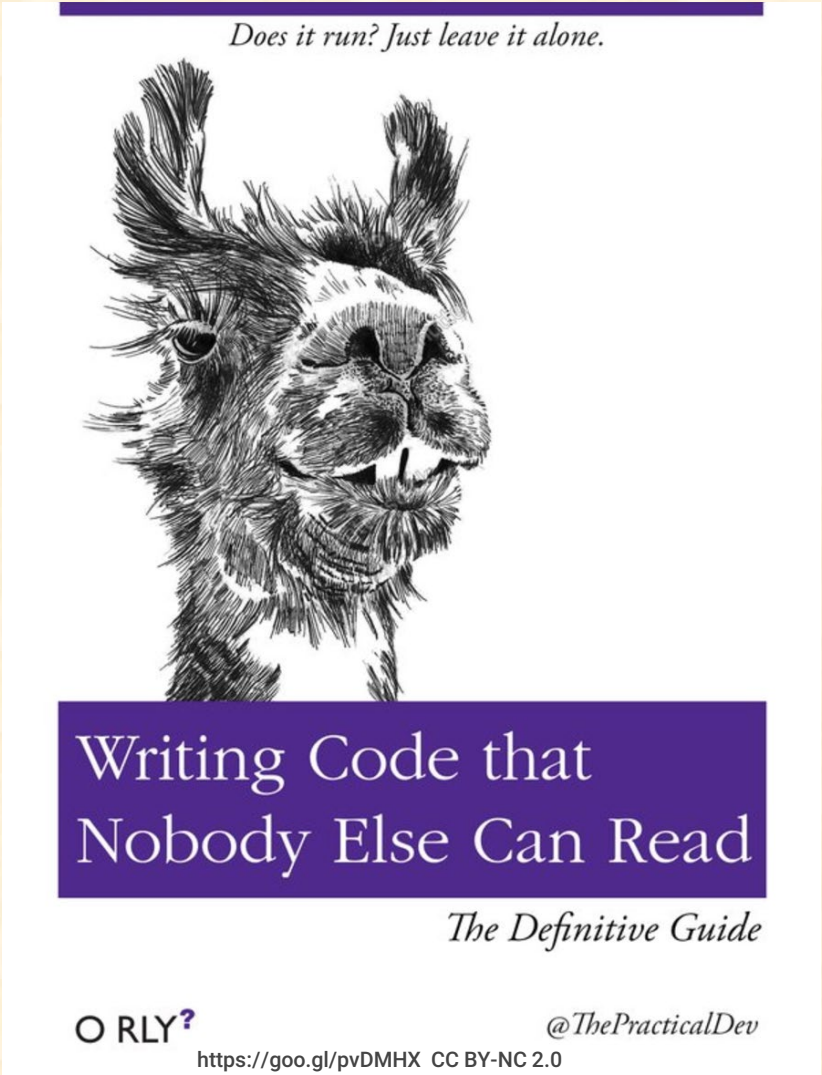
https://xkcd.com/1691/

- 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

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 problem
  - 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

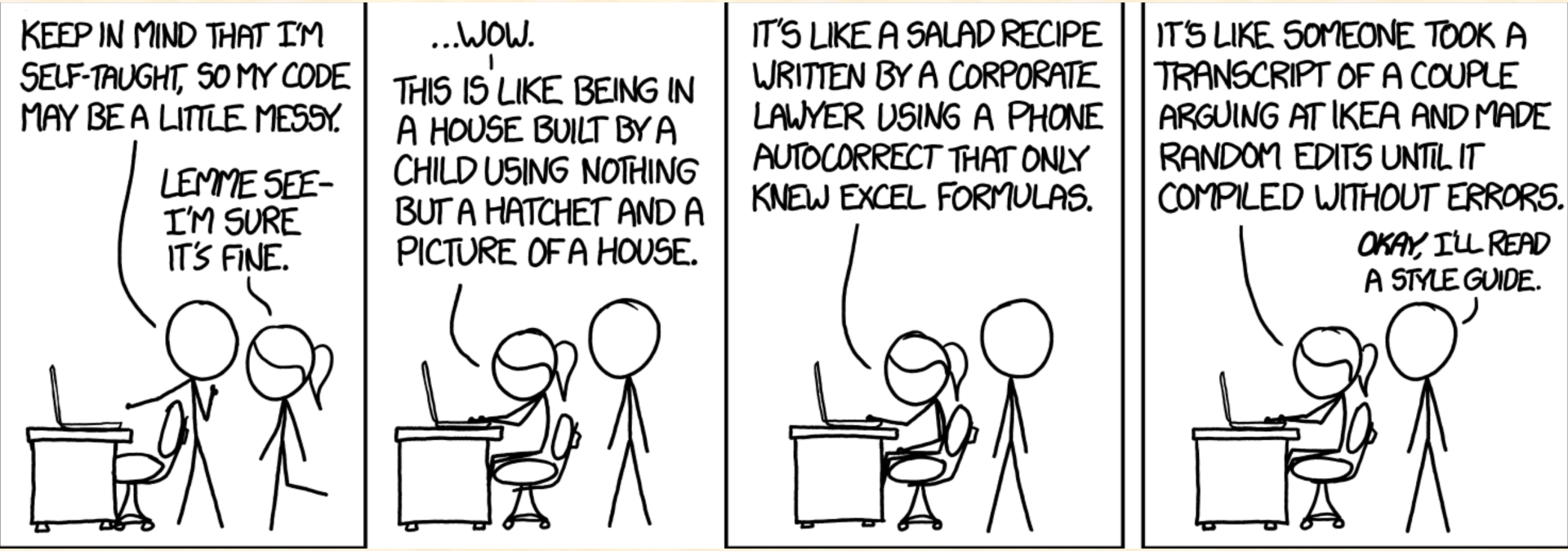
Great style depends upon point of view.



"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)



https://xkcd.com/1513/

