Tricky Code

To what degree should your code be concise instead of comprehensive? Will there always be a trade-off? Find out in the lesson below.

As a general rule I prefer explicit code to using “tricks” to make code extremely short. There is a trade-off, of course. Short, tricky code is less code, but the tradeoff is that it may be more difficult to understand by others, and even by yourself, when you return to the code after some period of time. Finding the optimal tradeoff between concise and clear is something you should be thinking about, and something that will be different for everyone.

If you want to see some examples of ridiculously “tricky” code, there is a competition called The International Obfuscated C Code Contest. Here is an example, from the 2011 IOCCC Competition, the entry called eastman by Peter Eastman:

```
#include <stdio.h>
#include <math.h>
#include <unistd.h>
#include <sys/ioctl.h>
            main() {
         short a[4];ioctl
      (0,TIOCGWINSZ,&a);int
    b,c,d=*a,e=a[1];float f,g,
  h,i=d/2+d%2+1,j=d/5-1,k=0,l=e/
 2,m=d/4,n=.01*e,o=0,p=.1;while (
printf("\x1b[H\x1B[?25l"),!usleep(
79383)){for (b=c=0;h=2*(m-c)/i,f=-
.3*(g=(l-b)/i)+.954*h,c<d;c+=(b=++
b%e)==0)printf("\x1B[%dm ",g*g>1-h
*h?c>d-j?b<d-c||d-c>e-b?40:100:b<j
||b>e-j?40:g*(g+.6)+.09+h*h<1?100:
 47:((int)(9-k+(.954*g+.3*h)/sqrt
  (1-f*f))+(int)(2+f*2))%2==0?107
    :101);k+=p,m+=o,o=m>d-2*j?
      -.04*d:o+.002*d;n=(l+=
         n)<i||l>e-i?p=-p
             ,-n:n;}}
```

You can compile and run it with the following two commands:

```
gcc -lm eastman.c -o eastman
./eastman
```

For more information on `gdb` and `valgrind`, there’s a comprehensive list of articles

For more information on `gdb` and `valgrind`, there's a comprehensive list of articles we've made in the lesson ahead.

Another important aspect of judging a code's efficiency is its runtime. For our next section, let's find ways to make our code faster.