



# Printing floating point numbers

## IV. Postlude

The standard C library never seems to do quite what you want for printing floats. If you want scientific notation, you can use "%e", but then 0 prints as 0.000000e+00. Or you can use %f, but then large numbers yield long strings of digits rather than the scientific notation you'd prefer.

As a parting gift, here's a routine that prints real numbers a little more nicely, automatically adjusting format codes depending on what kind of number you give it. You can specify how big or small a number can get before moving to scientific notation, and you can still specify field widths as in the usual "%n.nf" format.

```

1  #include <ieee754.h>
2  #define LOG2_10 3.321928095
3
4  #define flt_zero(x) (fabs(x) < EPSILON)
5
6  int max_digs_rt = 3; /* maximum # of
7                      scientific notation digits */
8  int max_digs_lf = 5; /* max # of digits before decimal point */
9
10 void print_real(double r, int width, int dec)
11 {
12     int mag;
13     double fpart, temp;
14     char format[8];
15     char num_format[3] = {'l', 0, 0};
16     union ieee754_double *dl;
17
18     dl = (union ieee754_double*)&r;
19     mag = (dl->ieee.exponent - IEEE754_1023) * LOG2_10;
20     if (r == 0)
21         mag = 0;
22     if ((mag > max_digs_lf-1) || (mag < -max_digs_rt))
23         num_format[1] = 'e';
24         temp = r/pow(10, mag); /*
25         fpart = temp - floor(temp); /*
26     }
27     else {
28         num_format[1] = 'f';
29         fpart = r - floor(r);
30     }
31     if (flt_zero(fpart))
32         dec = 0;
33     if (width == 0) {
34         snprintf(format, 8, "%%.%d%s",
35     }
36     else {

```

```
37     snprintf(format, 8, "%%d.%d%s\n", r, r, r);
38 }
39 printf(format, r);
40 }
```

[Advertising](#) | [Privacy policy](#) | [Copyright © 2019 Cprogramming.com](#) | [Contact](#) | [About](#)

