

Re: Computer programmers' habits in electronics

Source: <http://sci.tech-archive.net/Archive/sci.electronics.design/2005-12/msg03821.html>

- *From:* Spehro Pefhany <speffSNIP@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Wed, 21 Dec 2005 17:47:36 -0500
-

On Wed, 21 Dec 2005 12:28:17 -0800, the renowned Tim Wescott <tim@xxxxxxxxxxxxxxxxxxxx> wrote:

>>>
>Someone who likes to ask that question told me of an interviewee who got
>as far as saying "I think it involves recursion...".
>
>So I wrote a version that did it using recursive function calls and sent
>it to her. I don't know if I would have gotten the job --- they had
>already made the mistake of hiring me.

Something like this? (ugh)

```
#include <string.h>

rev2(char * istring, int start, int fin)
{
  char t;
  t = istring[start];
  istring[start] = istring[fin];
  istring[fin] = t;
  start++; fin--;
  if (fin > start) rev2(istring, start, fin);
}

....

/* reverse the string in place */
rev2(mystring, 0, strlen(mystring)-1);

....
```

>Good thing the PC has a lot of stack space.
>
>For interviewing embedded SW engineers we finally settled on a fairly
>basic scaling problem. We started with a little story problem that
>required the interviewee to find the ratio of a couple of numbers and
>multiply it to a third, then we said "oh, by the way, our floating point

Re: Computer programmers' habits in electronics

>library is too slow — do it with integers". The question usually took
>about 40 minutes to explore fully, with some folks never getting it and
>some just glancing at the board and writing down the correct answer.
>
>It's amazing how you can separate the desktop programmers from the
>embedded engineers with that one.

Here's one I did to approximate e^{-x} over the range 0 .. 0.75
(the end product would be written in assembler**, the C is just to
explain).

where xf, yf are 32-bit integers.

```
xf = x * 128;
yf = 32768 - xf + (xf* xf)/65536 - (((xf * xf)/32768
)*xf)/(6*32768) + (((xf*xf)/32768)* ((xf*xf)/32768))/(32768 * 24)
- (((((xf*xf)/32768)* ((xf*xf)/32768))/32768) * xf)/(32768 * 120)
;
```

The result is $\sim \exp(-x/256) * 32768$ over the range
x = 1 to 192.

yf/32768 agrees with $\exp(-x/256)$ within 0.001 over that range.

You can easily extend this to additional terms, although it gets a bit
tedious.

It's based on the textbook Taylor/Maclaurin series

infinity

\

$\exp(x) = \sum_{n=0}^{\infty} (x^n)/n!$

/

n= 0

** actually assembler macros, but that's another story

Best regards,
Spehro Pefhany

Re: Computer programmers' habits in electronics

"it's the network..." "The Journey is the reward"

speff@xxxxxxxxxxxxx Info for manufacturers: <http://www.trexon.com>

Embedded software/hardware/analog Info for designers: <http://www.speff.com>

- *Follow-Ups:*

- ◆ **Re: Computer programmers' habits in electronics**

- ◇ *From:* Mike Young

- ◆ **Re: Computer programmers' habits in electronics**

- ◇ *From:* Rich Grise

- *References:*

- ◆ **Re: Computer programmers' habits in electronics**

- ◇ *From:* onehappymadman

- ◆ **Re: Computer programmers' habits in electronics**

- ◇ *From:* Rich Grise, but drunk

- ◆ **Re: Computer programmers' habits in electronics**

- ◇ *From:* Rich Grise, but drunk

- ◆ **Re: Computer programmers' habits in electronics**

- ◇ *From:* Tim Wescott

- Prev by Date: **Re: Computer programmers' habits in electronics**

- Next by Date: **Re: Rotary encoder**

- Previous by thread: **Re: Computer programmers' habits in electronics**

- Next by thread: **Re: Computer programmers' habits in electronics**

- Index(es):

- ◆ **Date**

- ◆ **Thread**