

# Re: Recursive functions

---

*Source:* [http://coding.derkeiler.com/Archive/C\\_CPP/comp.lang.c/2007-04/msg00129.html](http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-04/msg00129.html)

---

- *From:* "Bill Pursell" <[bill.pursell@xxxxxxxx](mailto:bill.pursell@xxxxxxxx)>
  - *Date:* 1 Apr 2007 14:07:46 -0700
- 

On Apr 1, 9:24 pm, "Bill Pursell" <[bill.pursell@xxxxxxxx](mailto:bill.pursell@xxxxxxxx)> wrote:

On Apr 1, 9:10 pm, Lauri Alanko <[l...@xxxxxx](mailto:l...@xxxxxx)> wrote:

In article <1175456107.493063.221...@xx>,

Bill Pursell <[bill.pursell@xxxxxxxx](mailto:bill.pursell@xxxxxxxx)> wrote:

Because it is totally inappropriate to use a recursive function to compute the length of a string.

"Totally inappropriate" apparently means here "inefficient on a typical C implementation". That is certainly true, but not always crucial. Even the space performance isn't an issue if the lengths of all argument strings are known to have a reasonable bound.

Efficiency aside, recursion is certainly a natural way of defining the length of a sequence.

It is a natural way of defining the length of a finite sequence in a mathematical setting. It is not completely unnatural to define the length of a finite sequence in terms of recursion on a computer. However, it IS totally inappropriate to compute the value using recursion. Totally inappropriate does not mean "inefficient in a typical C implementation". It is, rather, a euphemism for "completely boneheaded".

I should probably learn to pause for at least 10 seconds before posting, if only to catch stupid spelling errors. To clarify why I think it's wrong to use recursion in

## Re: Recursive functions

this case: things should be kept as simple as possible, but no simpler. The standard library provides `strlen`, and `"return strlen( a );"` is simpler than `"return 1 + strlen( a + 1 );"`. I'm actually not at all concerned about efficiency of the implementation, and in fact it wouldn't bother me if `strlen` were implemented as a recursive function. Well, maybe not. :)

.