

Re: Fortran vs. C (was Re: Ken & Klaus.)

Source: <http://sci.tech-archive.net/Archive/sci.physics/2004-09/8149.html>

From: Harry Conover (hhc314_at_yahoo.com)

Date: 09/21/04

Date: 20 Sep 2004 17:50:36 -0700

nessuno@wigner.berkeley.edu (nessuno) wrote in message
news:<f76c1166.0409201050.1d7bd2b@posting.google.com>...
> *beliavsky@aol.com* wrote in message news:<3064b51d.0409180938.366bc3f2@posting.google.com>...
>> *nessuno@wigner.berkeley.edu* (nessuno) wrote in message
news:<f76c1166.0409170717.6ebe4e11@posting.google.com>...
>>> *Robert Newson* <ReapNewsB@bullet3.fsnet.oc.ku> wrote in message
news:<414ABB87.7070100@bullet3.fsnet.oc.ku>...
>>>> *Jeff Relf* wrote:
>>>>>
>>>>
>>>> <snip>
>>>>
>>>> *I did a lot of Fortran programming, most of it Fortran 66 (!). The
>>>> older Fortrans were pretty awful by today's standards of what a
>>>> programming language should be. One problem in upgrading them was
>>>> that you had to introduce incompatibilities with previous versions to
>>>> change some things. For example, on going from Fortran 66 to Fortran
>>>> 77, loop testing was moved to the start of the loop (in old Fortran,
>>>> all loops would execute at least once, unless you explicitly put in a
>>>> test to branch around them). I never learned Fortran 95, by then I
>>>> had switched to C. But it was obvious that in order to improve
>>>> Fortran 77, one would have to introduce further incompatibilities with
>>>> it.*
>>>>
>>>> *What seems "obvious" to you is incorrect in this case. Fortran 90
>>>> greatly improved on Fortran 77 and maintained full backward
>>>> compatibility. Every standard-conforming Fortran 77 program is an F90
>>>> program, with the same meaning.*
>>>>
>>>>> *Once you've crossed this threshold, the question is, how far do
>>>>> you go? If you're willing to go far enough in introducing
>>>>> incompatibilities, you could transform Fortan exactly into C, maybe in
>>>>> several steps if you wanted to do that. To me it seemed pointless.*
>>>>> *Why not just use C?*
>>>>>
>>>>> *Because C is missing a lot of Fortran 95 features:*
>>>>>
>>>>> (1) *In Fortran 95, you can allocate a 3-D array with just*

>>
>> *real, allocatable :: x(:, :, :)*
>> *allocate (x(n1, n2, n3))*
>>
>> *The corresponding C code is longer and trickier.*
>>
>> *(2) Passing arrays to functions is easier, since their size can*
>> *be queried. In Fortran you can just write*
>>
>> *call foo(x)*
>>
>> *and start subroutine foo with*
>>
>> *subroutine foo(x)*
>> *real, intent(in) :: x(:, :, :)*
>> *integer :: n1, n2, n3*
>> *n1 = size(x, 1)*
>> *n2 = size(x, 2)*
>> *n3 = size(x, 3)*
>>
>> *It's not easy to write a C function taking a general 2-D or 3-D array*
>> *AFAIK.*
>>
>> *(3) Fortran 90/95 has array operations and array sections comparable*
>> *to Matlab, letting you write fewer lines of code which better reflect*
>> *the operations you are trying to perform.*
>>
>> *(4) When you do need loops, the Fortran DO-loop is safer than the for*
>> *loop of C because you are guaranteed that the loop variable does not*
>> *change within the loop. Usually this is what I want. If not, a*
>> *DO-ENDDO loop without a loop variable but with an EXIT suffices.*
>>
>> *(5) Until C99, which is not so widely used as C90, there are no*
>> *complex variables in C.*
>
> *Yeah, I forgot to mention, original C didn't have complex variables,*
> *and it would have hardly been difficult to have included them from the*
> *beginning. And it's a pain to work around this.*
>
> *Maybe if I had to do serious scientific programming again, I'd have to*
> *learn Fortran 95, but I'm not enthusiastic about it.*
>
> *Here is a question: Is there anything a general purpose programming*
> *language (like C) needs that a scientific programming language doesn't*
> *need? I don't think so, especially if you want to write large*
> *programs. The converse is not true, of course, there are special*
> *needs for scientific programs, like complex variables, and efficiency*
> *is paramount. So in an ideal world, we'd have just one language for*
> *both scientific programming and everything else.*
>
> *Sounds like from what you have above that Fortran 95 passes length*

sci.physics: Re: Fortran vs. C (was Re: Ken & Klaus.)

- > *arguments along with array pointers. Sort of like what PL/I used to*
- > *do with strings (they were called "dope vectors", i.e., the structures*
- > *containing the length and pointer to the actual string).*
- >
- > *If Fortran 95 is backward compatible, then it keeps the record*
- > *oriented I/O, which I find very awkward after working with C.*
- >
- > *Nessuno*

Amazing, stupidly I hadn't realized that Fortran was still used, given that the last time I saw it was back around 1970! :-) Still, as a physicist I always like it as a turn-the-crank number cruncher.

Seriously speaking though, I really don't understand how anyone familiar with both Fortran and C can compare the two without cracking a smile. Fortran is a high level application oriented language, while C is a relatively low-level systems implementation language.

Comparing the two is like comparing a cash register to a traffic light (bad analogy I know, but it is the best that I could come up with on short notice). Still, I believe you get my point.

Comparing Fortran to C is in my mind similar to comparing Fortran to Assembly Language, since C is only a slight step above assembly language. C offsets this limitation by providing absolutely wonderful data structuring and access shortcuts called (strangely enough) pointers and structures. By contrast, navigating your way through complex data structures in Fortran becomes a major hassle, and a great burner of CPU time.

Then too, since each of these languages have a role in different markets, they are rarely placed in a position of competing with one another.

Just my thoughts...

Harry C.