

## Re: operation on module arguments

---

*Source:* <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2006-09/msg00820.html>

---

- *From:* Brooks Moses <[bmoses-nospam@xxxxxxxxxxxxxxxxxxxxx](mailto:bmoses-nospam@xxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Thu, 21 Sep 2006 17:48:23 -0700
- 

gottoomanyaccounts@xxxxxxxxx wrote:

From Richard Maine and other people, I understand now there is no easy way to do what I want. But the logic to have this feature in the language is pretty reasonable, so I was not too greedy, was I ? :)

When "use module" appears in any program unit, compiler knows it at compilation time. Then during execution time, whenever variables defined in the module that has been used here are modified, the rest of the variables defined in that same module should get updated automatically, upon the relation also defined in the same module, if that relation exists. Otherwise why we bother to "use module"? By "use" it, we really want to use all information in there.

The key part of this is "if that relation exists". It cannot exist within the Fortran language. Thus, it's quite obvious why "use module" doesn't transfer such things.

So, essentially, what you're doing is asserting (implicitly) that the ability to express such a relation in Fortran should exist. I have no idea if your logic for why it should exist is reasonable or not, since you haven't said anything about why it should exist. :)

Does all of this even make sense to implement in the language? How many people would appreciate this feature? :P

The thing is that what "real A" means is "allocate a certain bit of memory for storing a real variable, and let the variable name 'A' refer to that bit of memory."

You're asking for it to potentially mean, instead, "allocate a certain bit of memory for storing a real variable, and let any attempts to update the variable name 'A' invoke a set of procedures as well as referring to this bit of memory." (But it would only mean this if something was defined in relation to it.)

That would mean, fundamentally, that A is no longer acting like a real variable. It means, for example, that the compiler can't pass A around like a real variable, because when the compiler passes it to a subroutine that might modify it, the compiler also has to pass along the instructions that go with it. Also, it means that it can't treat it like a real variable in code optimizations.

## Re: operation on module arguments

Furthermore, you're asking for it to change its definition when C is declared, not when A is declared. This seems like quite a confusing idea; either there are annoying strict rules about A and C having to be in exactly the same scope (as opposed to importing A from a different module, say), or it will have all sorts of confusing side effects about when C is or is not updated.

Finally, what you're asking for is very little different from "whenever the value of C is requested, it is calculated from A and B." This concept already exists in the language — it's called a function. Is there really a need for something different to do the same thing?

– Brooks

—

The "bmoses-nospam" address is valid; no unmunging needed.

.