

Re: Windows array allocation problem

Source: <http://coding.derkeiler.com/Archive/Fortran/comp.lang.fortran/2007-09/msg00967.html>

- *From:* nospam@xxxxxxxxxxxxxx (Richard Maine)
 - *Date:* Sat, 29 Sep 2007 10:24:40 -0700
-

James Van Buskirk <not_valid@xxxxxxxxxxxx> wrote:

What happens when a bound gets over 32 bits

Its a hassle (unless default integer is 64 bits). Yes, you have to use a 64-bit integer kind (or, I suppose a 128-bit one, but that's rare) for any value that exceeds the range of 32-bit integers, but it isn't quite as bad as your question s indicate.

I suppose the bound given in the ALLOCATE statment has to be a 64-bit variable of constant.

Well, sort of. No magic requirement here – just the "obvious". If a 32-bit integer can't represent the value being used, then you can't use a 32-bit integer for that value. Note that this tautology does not extend to any more esoteric requirement. It pretty much is just the tautology. For example, just because one bound won't fit in 32 bits and thus needs a larger kind, that doesn't say anything about the other bounds.

Does this mean that any subsequent array reference has to have a 64-bit index or array subscript triplet?

No. Absolutely not. Thats' important. It is perfectly fine to use a 32-bit integer... or heck, an 8-bit one if the particular value happens to fit in one. There is not a requirement for kind agreement here. The only requirement is the tautological one – that if a paticular value won't fit in a particular kind of integer, then you can't use that kind of integer for that value.

So if you have a variable that is being used as an index and can range over the entire range of index values, you better use a 64-bit integer fr that variable. But if the particular variable is only used for index

Re: Windows array allocation problem

values that fit it 32-bits, the variable can still be a 32-bit integer.

So this means that in many/most cases, yes, you will need index variables to be 64-bit ones in practice. But that isn't a special requirement.

Can an automatic array go over the top? Can an array expression do this? Can an elemental procedure reference do this? What happens when an assumed shape array