Subject: Re: User selectable lower array bound?
Posted by Paul van Delst on Thu, 09 Aug 2001 13:15:46 GMT
View Forum Message <> Reply to Message

Jeff Guerber wrote:
>
> On 3 Aug 2001 bennetsc@NOSPAMucs.orst.edu wrote:
>
>>> It seems like such a simple thing to be able to do (with default
>>> action being start at
>>> index 0) although I'm sure the amount of work required to
>>> implement this would be
>>> horrendous. Still, it shur would be nice.....
>>>
>> That depends upon how IDL already keeps track of arrays
>> internally. In PL/1, for example, one declared an array with the
>> boundaries for each dimension in the form lowerbound:upperbound,
>> where specification of the lower bound and the colon were optional.
>> If only the upper bound were specified, then the lower bound defaulted
>> to 1. In its internal representation of arrays, IIRC, PL/1 kept
>> the lower and upper boundaries of each dimension as part of a control
>> block preceding the actual array memory. If a language implementation
>> doesn't already store both boundaries, or equivalently, the lower
>> boundary and number of elements, for each dimension, then yes, adding
>> such support might well be a major headache.
>
> Well, IDL does perform bounds checking, even for arrays passed into a
> procedure as arguments, so it must already store at least either the upper
> bound or the number of elements (which are equivalent since the lower
> bound is fixed). It's likely that this is only done in one place, so
> implementing lower bounds in the IDL core might not be all _that_ much
> work. HOWEVER...
>
> Having thought about this further, I now think the more serious problem
> would be all the library procedures (and not just RSI's!) that assume you
> can loop over the elements of any array by going from 0 to
> n_elements(array)-1. (Aiiigh!) Unless the bounds are lost across
> procedure calls (as Paul pointed out that Fortran does), which can
> sometimes be useful but which kind of defeats the point of having
> definable bounds, if you ask me.

Most definitely. There has to be a way of defining the bounds across routine calls. I like the syntax that Scott Bennet suggested:

```
my_array( -10:10 )
```

or

```
  my_array[ -10:10 ]
```

or something like that. If there an array like x=FLTARR(10), passing "x" should be the same as passing x[0:9] if we didn't have to deal with the bloody silly pass by reference or pass by value problem.

O.k. now it's my turn...

<rant>
That is one beef I have with IDL - that fact that I can't do something like

```
  x = FLTARR( 10, 10 )

  for i = 0, 9 do begin
    result = my_complicated_func( x[ *, i ] )
  endfor
```

and have the slices of x filled up as it goes instead of

```
  for i = 0, 9 do begin
    result = my_complicated_func( dummy_x )
    x[ *, i ] = dummy_x
  endfor
```

Or, even worse, something like:

```
  x = FLTARR( 10, 10 )
  openr,1,'my_file_of_numbers'

  for i = 0, 9 do begin
    readu, 1, x[ *, i ]
  endfor
```

rather than

```
  for i = 0, 9 do begin
    readu, 1, dummy_x
    x[ *, i ] = dummy_x
  endfor
```

Please remember these are very simple examples.

The online help even states it's an awkward interface: (From "Parameter Passing Mechanism")

"The correct, though somewhat awkward, method is as follows:

```
 TEMP = ARR[5]
```

ADD, TEMP, 4
ARR[5] = TEMP"

I think it's silly - at least nowadays - that the user has to even consider *how* the variables are passed, i.e. by reference or value. I sure don't care and having to declare dummy arrays for purposes like the above just bugs me. IDL was created out of/from (?) F77 which passed all arguments one way or another (can't remember which.) Fortran compilers nowadays do it either way based on what optimises better.
</rant>

Not having to bother about reference or value argument passing would maybe clear the way to allowing the passage of arbitrarily bounded arrays like:

```
result = my_func( x[-10:20,*] )
```

so that in "my_func" the code recognises the specified lower and upper bounds on the first array index. If one simply did:

```
result = my_func( x )
```

even if x was declared with bounds [-10:20, 0:whatever], the function my_func would see the argument as a 2-D array with bounds of [0:31,0:whatver].

But I agree with Jeff in that making this foolproof for all the existing code would be a [CAUTION: understatement ahead] Pretty Big Task. You'd have to create an IDL function that checked the lower and upper bounds, insert that in all the relevant code/functions/procedures and then make sure that the lower bound == 0 and the upper one == n_elements(array)-1. Oof. A soul destroying task at best (any grad students out there volunteer to intern at RSI for, oh I don't know, a couple of years..?) But, that's what shell scripts and sed are for....

Having said all that I still think IDL is one of the much better things that have come to pass since sliced bread. :oD I'd be lost without it.

paulv

p.s. I *was* just kidding about the script/sed thing.....

--
Paul van Delst A little learning is a dangerous thing;
CIMSS @ NOAA/NCEP Drink deep, or taste not the Pierian spring;
Ph: (301)763-8000 x7274 There shallow draughts intoxicate the brain,
Fax:(301)763-8545 And drinking largely sobers us again.
Alexander Pope.