Subject: User selectable lower array bound?
Posted by Paul van Delst on Thu, 02 Aug 2001 21:47:03 GMT
View Forum Message <> Reply to Message

Hey there,

Is is just me, or would anyone else find useful the ability to define arrays in IDL such that the lower bound is *not* always zero? Sorta like:

```
  x = FINDGEN( 11, LOWER = -5 )
or
  y = DBLARR( 100, LOWER = 1 )
```

so that accessing elements such as x[-4] or y[100] are o.k.?

I know this can be done now with judicious use of proxy indices, e.g.

```
  FOR i = -5, 5 DO BEGIN
    ix = i + 5
    PRINT, x[ ix ]
    ....do other stuff with negative i's....
  ENDFOR
```

but sometimes this makes code hard to follow (or explain to someone who's never used the code before) in direct correspondence with a physical process.

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.....

paulv

--
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.

Subject: Re: User selectable lower array bound?
Posted by Marcus O'Brien on Fri, 03 Aug 2001 10:40:10 GMT
View Forum Message <> Reply to Message

Yep,

This is something I wanted to use for converting images to center of mass coordinates for principle axes transformations, ended up faffing around to simulate it. The concept does

appear in display coordinate systems for objects though (well it seems vaguely related to me).

Marc

Paul van Delst wrote:

> Hey there,
>
> Is is just me, or would anyone else find useful the ability to define arrays in IDL such
> that the lower bound is *not* always zero? Sorta like:
>
> x = FINDGEN(11, LOWER = -5)
> or
> y = DBLARR(100, LOWER = 1)
>
> so that accessing elements such as x[-4] or y[100] are o.k.?
>
> I know this can be done now with judicious use of proxy indices, e.g.
>
> FOR i = -5, 5 DO BEGIN
> ix = i + 5
> PRINT, x[ix]
> ....do other stuff with negative i's....
> ENDFOR
>
> but sometimes this makes code hard to follow (or explain to someone who's never used the
> code before) in direct correspondence with a physical process.
>
> 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.....
>
> paulv
>
> --
> 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.

Subject: Re: User selectable lower array bound?
Posted by Stein Vidar Hagfors H[1] on Fri, 03 Aug 2001 16:25:59 GMT
View Forum Message <> Reply to Message

Paul van Delst <paul.vandelst@noaa.gov> writes:

[..]
> Well, maybe WHERE could work as it does now, but for cases where the start index is not
> zero, a function like the Fortran 90 intrinsic LBOUND() could be used.
>
> BTW, I never check the WHERE result either, always the COUNT value.

Funny, I [almost] never use the COUNT value.. If you're writing an IF
test, the information is there anyway..

```
 ix = where(nonzero)
 IF ix[0] ne -1 THEN ...
```

--

 -- --------------
Stein Vidar Hagfors Haugan
ESA SOHO SOC/European Space Agency Science Operations Coordinator for SOHO

NASA Goddard Space Flight Center, Email: shaugan@esa.nascom.nasa.gov
Mail Code 682.3, Bld. 26, Room G-1, Tel.: 1-301-286-9028/240-354-6066
Greenbelt, Maryland 20771, USA. Fax: 1-301-286-0264
 -- --------------

Subject: Re: User selectable lower array bound?
Posted by Paul van Delst on Fri, 03 Aug 2001 16:49:04 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
>
> Paul van Delst <paul.vandelst@noaa.gov> writes:
>
> [..]
>> Well, maybe WHERE could work as it does now, but for cases where the start index is not
>> zero, a function like the Fortran 90 intrinsic LBOUND() could be used.
>>
>> BTW, I never check the WHERE result either, always the COUNT value.
>
> Funny, I [almost] never use the COUNT value.. If you're writing an IF
> test, the information is there anyway..
>
> ix = where(nonzero)
> IF ix[0] ne -1 THEN ...

Sure. It's just a matter of personal preference.

This stuff fits into the "same dog different leg" category. :o)

paulv

--
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.

Subject: Re: User selectable lower array bound?
Posted by Craig Markwardt on Fri, 03 Aug 2001 22:08:46 GMT
View Forum Message <> Reply to Message

Paul van Delst <paul.vandelst@noaa.gov> writes:

> Hey there,
>
> Is is just me, or would anyone else find useful the ability to define arrays in IDL such
> that the lower bound is *not* always zero? Sorta like:
>
> x = FINDGEN(11, LOWER = -5)
> or
> y = DBLARR(100, LOWER = 1)
>
> so that accessing elements such as x[-4] or y[100] are o.k.?

Well, as grumpy as I have been in the past about IDL wishes, this is one thing I do not want to have in IDL now!

As has been pointed out, the issue gets clouded when dealing with things like WHERE() [btw, <flame>I think that -1 is one of the single biggest mistakes that IDL has in it<flame>], but also about how such an array would be passed to other procedures. Also, IDL has other ways of indexing arrays, like indexing a 2d array by 1d subscripts. What would happen then?

For the most part, I find it pretty easy to bite the bullet and say x[CONST-4] instead of x[4].

Which brings a gripe of my own to mind. One tricky problem with accessing arrays is that accessing out of bounds elements is both legal and illegal, depending on whether the subscript index is a scalar or an array.

```
a[ [-1] ] = 0   ;; Succeeds!
a[  -1  ] = 0   ;; Fails!
```

What is up with that? :-)

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: User selectable lower array bound?
Posted by John-David T. Smith on Fri, 03 Aug 2001 22:36:07 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:

> Which brings a gripe of my own to mind. One tricky problem with
> accessing arrays is that accessing out of bounds elements is both
> legal and illegal, depending on whether the subscript index is a
> scalar or an array.
>
> a[[-1]] = 0 ;; Succeeds!
> a[-1] = 0 ;; Fails!
>
> What is up with that? :-)

From the manual:

>>>> >>>>>>>>>>
Using Arrays as Subscripts:
<snip>

If an element of the subscript array is less than or equal to zero, the first element of the subscripted variable is selected. If an element of the subscript is greater than or equal to the last subscript in the subscripted variable (N, above), the last element is selected.
<<<<<<<<<<<<<

Probably a user complained about having to check the bounds of their array of indices, and RSI added this feature without thinking about the consistency ramifications... apparently a theme of theirs.

JD

Subject: Re: User selectable lower array bound?
Posted by Jeff Guerber on Fri, 03 Aug 2001 23:11:20 GMT
View Forum Message <> Reply to Message

On 3 Aug 2001, Stein Vidar Hagfors Haugan wrote:

> Paul van Delst <paul.vandelst@noaa.gov> writes:
>
> [..]
>> Well, maybe WHERE could work as it does now, but for cases where the start index is not
>> zero, a function like the Fortran 90 intrinsic LBOUND() could be used.
>>
>> BTW, I never check the WHERE result either, always the COUNT value.
>
> Funny, I [almost] never use the COUNT value.. If you're writing an IF
> test, the information is there anyway..
>
> ix = where(nonzero)
> IF ix[0] ne -1 THEN ...

Stein Vidar, the point I was making about WHERE was that with definable lower bounds, -1 could be a legitimate index into the array, but consistently using COUNT avoids this problem. I was also speculating that WHERE returning lbound-1 would be consistent with its current behavior, and wouldn't break (as many) existing programs, in which the arrays have an implicit lbound=0 (and thus lbound-1 = -1).

(Actually, I've been known to use either form, depending on how I feel at the time. :-) I've been trying to use COUNT more, though.)

There are other problems, though. <sigh>

Jeff Guerber
Raytheon ITSS
NASA Goddard Space Flight Ctr
Oceans & Ice Branch (code 971)

Subject: Re: User selectable lower array bound?
Posted by thompson on Mon, 06 Aug 2001 14:29:23 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:

> Paul van Delst <paul.vandelst@noaa.gov> writes:

>> Hey there,
>>
>> Is is just me, or would anyone else find useful the ability to define arrays in IDL such
>> that the lower bound is *not* always zero? Sorta like:
>>
>> x = FINDGEN(11, LOWER = -5)
>> or
>> y = DBLARR(100, LOWER = 1)
>>
>> so that accessing elements such as x[-4] or y[100] are o.k.?

> Well, as grumpy as I have been in the past about IDL wishes, this is
> one thing I do not want to have in IDL now!

I'm glad I'm not the only one! When I first saw this proposal, I thought to myself that I could hear the tinkly sound of code breaking all over the place. :^)

William Thompson

Subject: Re: User selectable lower array bound?
Posted by Paul van Delst on Mon, 06 Aug 2001 16:06:51 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Ben Tupper writes:
>
>> I just whipped up an object to allow you to do just this kind of
>> pseudo-indexing for 1d vectors.
>
> Oh, sure, you can do it with objects. But
> isn't that, like, cheating? In any case,
> scientists are too busy to learn yet one
> more useful programming practice.

Mostly, yes, :o) but, with all due respect to Ben and his rather natty and cool code snippet (I'm facing north towards Maine, bowing :o), I would rephrase it more like:

"some scientists would like variable-bounds indexing to be indistinguishable from current command line vector/matrix operations without having to learn :

;--------START HERE
; EXAMPLE

```
;       IDL> x = obj_new('findgen', 11, lower = -5)
;       IDL> print, x->GetData([-4, 0, 5])
;      1.00000      5.00000      10.0000
;       IDL> obj_destroy, x
;------
;       GetData
;------
FUNCTION FINDGEN::GetData, Indices
Return, (*Self.Data) [Indices - Self.Lower]
END; GetData
;------
;       SetProperty
;------
PRO FINDGEN::SetProperty, N = N, Lower = Lower
If n_elements(Data) NE 0 Then *Self.Data = Data
If n_elements(N) NE 0 Then Begin
                ;1d vectors only
        Self.N = N[0]
                ;either 'redefine' the variable or 'undefine' it
        If Self.N GT 0 Then *Self.Data = Findgen(Self.N) Else $
                Dummy = Size(Temporary(*Self.Data))
EndIf
If n_elements(Lower) NE 0 Then Self.Lower = Lower[0]
END     ;SetProperty
;------
;       GetProperty
;------
PRO FINDGEN::GetProperty, Data = Data, N = N, Lower = Lower
If Arg_present(Data) Then Data =  *Self.Data
N = Self.N
Lower = Self.Lower
END     ;GetProperty
;------
;       Initialization
;------
FUNCTION FINDGEN::INIT, N, LOWER = lower
If n_elements(N) EQ 0 Then Self.N = 0L Else Self.N = 0L > N[0]
If n_elements(lower) EQ 0 Then Self.Lower = 0L Else Self.Lower = Lower[0]
If Self.N GT 0 Then Self.Data = Ptr_NEW(Findgen(Self.N))        Else $
        Self.Data = Ptr_NEW(/Allocate)
Return, 1
END     ;Init
;-----
; CleanUp
;-----
PRO FINDGEN::CleanUp
If Ptr_Valid(Self.Data) Then Ptr_Free, Self.Data
END     ;CleanUp
```

```
;-----
;       Definiton
;------
PRO FINDGEN__DEFINE
Struct = {FINDGEN, $
        Data: ptr_new(), $      ; the data array
        N: 0L, $                  ;this handles only 1d arrays right now
        Lower: 0L}                  ;the indexed address of the lower bound
END
;--------END HERE"
```

--
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.

Subject: Re: User selectable lower array bound?
Posted by Pavel A. Romashkin on Mon, 06 Aug 2001 16:21:07 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> Well, as grumpy as I have been in the past about IDL wishes, this is
> one thing I do not want to have in IDL now!

I am with you Craig. Besides, for the purists of array indexing, I think it is unfair to dasignate a *lower* array bounds. We don't designate the *upper* one. To be exact, we need a zero point fixed and the ability to extend an array in both directions. This way, I can add data in both positive and negative directions. And Ben will be providing a code snippet here in a few minutes. Is that from your object programming book, by the way, Ben :-)?

Cheers,
Pavel

P.S. I think David needs not worry about scientists learning new useful techniques :-(

Subject: Re: User selectable lower array bound?
Posted by Paul van Delst on Mon, 06 Aug 2001 16:47:01 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:
>
> Craig Markwardt wrote:
>>
>> Well, as grumpy as I have been in the past about IDL wishes, this is
>> one thing I do not want to have in IDL now!
>
> I am with you Craig. Besides, for the purists of array indexing, I think
> it is unfair to dasignate a *lower* array bounds. We don't designate the
> *upper* one.

In the context of initially declaring an array in IDL, sure you do:

x = fltarr(10)

declares the upper bound as 9. We also designate a lower bound: 0. The difference between the two is that I can change the former.

> To be exact, we need a zero point fixed

why?

> and the ability to
> extend an array in both directions. This way, I can add data in both
> positive and negative directions.

Why would this functionality be any different to what exists now? And, to me at least, allowing -ve indices would make this sort of data manipulation easier to understand, i.e. extend array in -ve direction => negative indices.

paulv

> P.S. I think David needs not worry about scientists learning new useful
> techniques :-(

I agree. :o\

--
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.

Subject: Re: User selectable lower array bound?
Posted by John-David T. Smith on Mon, 06 Aug 2001 20:52:13 GMT

Paul van Delst wrote:
>
> "Pavel A. Romashkin" wrote:
>>
>> Craig Markwardt wrote:
>>>
>>> Well, as grumpy as I have been in the past about IDL wishes, this is
>>> one thing I do not want to have in IDL now!
>>
>> I am with you Craig. Besides, for the purists of array indexing, I think
>> it is unfair to dasignate a *lower* array bounds. We don't designate the
>> *upper* one.
>
> In the context of initially declaring an array in IDL, sure you do:
>
> x = fltarr(10)
>
> declares the upper bound as 9. We also designate a lower bound: 0. The difference between
> the two is that I can change the former.
>
>> To be exact, we need a zero point fixed
>
> why?
>
>> and the ability to
>> extend an array in both directions. This way, I can add data in both
>> positive and negative directions.
>
> Why would this functionality be any different to what exists now? And, to me at least,
> allowing -ve indices would make this sort of data manipulation easier to understand, i.e.
> extend array in -ve direction => negative indices.

The most annoying thing about IDL arrays to me is the need always to
test whether they exist or not when concatenating onto them. The idea
of extending arrays in both directions would be neatly summed up by
allowing:

a=[b,a] & a=[a,b] even if a doesn't (yet) exist.

Either that, or IDL needs a list type which allows such operations.
Wasn't that just me ranting about special case functionality leading to
inconsistency?

But anyway, my $2e-2.

JD

Subject: Re: User selectable lower array bound?
Posted by bennetsc on Tue, 07 Aug 2001 03:26:04 GMT
View Forum Message <> Reply to Message

In article <3B6F037D.B17A6287@astro.cornell.edu>,
JD Smith <jdsmith@astro.cornell.edu> wrote:
> Paul van Delst wrote:
>>
>> "Pavel A. Romashkin" wrote:
>>>
>>> Craig Markwardt wrote:
>>>>
>>>> Well, as grumpy as I have been in the past about IDL
> wishes, this is
>>>> one thing I do not want to have in IDL now!
>>>
>>> I am with you Craig. Besides, for the purists of array
> indexing, I think
>>> it is unfair to dasignate a *lower* array bounds. We don't
> designate the
>>> *upper* one.
>>
>> In the context of initially declaring an array in IDL, sure you do:
>>
>> x = fltarr(10)
>>
>> declares the upper bound as 9. We also designate a lower
> bound: 0. The difference between
>> the two is that I can change the former.
>>
    So how about if {flt,dbl,complex,int,lon,dcomplex,byt,str}arr
and make_array could accept both the form shown above and this form:

 y = fltarr(-5:10)

which would declare the lower bound as -5 and the upper bound as 9,
giving a total of 16 elements, including the zero element? This
isn't quite as nice as PL/1's method because of the zero element,
but it would be usable and wouldn't break any existing code. Future
programs would have to take into consideration that

 y = fltarr(-5:-1)

would have a lower bound of -5 and an upper bound of -1, giving
a total of only 5 elements due to the lack of a zero element. PL/1's
syntax avoided this problem by having the lower bound default to 1
if not coded, but I think I could live with it as long as I were
aware of it.

Scott Bennett, Comm. ASMELG, CFIAG
College of Oceanic and Atmospheric
Sciences
Oregon State University
Corvallis, Oregon 97331
********** **********
* Internet: sbennett at oce.orst.edu *
---------- ---------
* "Lay then the axe to the root, and teach governments humanity. *
* It is their sanguinary punishments which corrupt mankind." *
* -- _The_Rights_of_Man_ by Tom Paine (1791.) *
********** **********

Subject: Re: User selectable lower array bound?
Posted by marc schellens[1] on Tue, 07 Aug 2001 11:36:53 GMT
View Forum Message <> Reply to Message

> Think Perl. For example, to add an element or elements to the end of a
> possibly non-existent list, you say:
>
> push @list, $elem;
>
> To pull one element off the end:
>
> pop @list;
>
> To pull one off the front:
>
> shift @list;
>
> To push one onto the front:
>
> unshift @list, $elem;
>
> With such a collection we could rid ourselves of all those silly
> statements like:
>
> if n_elements(list) eq 0 then list=[elem] else list=[list, elem]

Ahem:

```
pro push,list,elem
if n_elements(list) eq 0 then list=[elem] else list=[list, elem]
end
```

cheers,

Subject: Re: User selectable lower array bound?
Posted by John-David T. Smith on Tue, 07 Aug 2001 15:24:58 GMT
View Forum Message <> Reply to Message

bennetsc@NOSPAMucs.orst.edu wrote:
>
> In article <3B6F037D.B17A6287@astro.cornell.edu>,
> JD Smith <jdsmith@astro.cornell.edu> wrote:
>> Paul van Delst wrote:
>>>
>>> "Pavel A. Romashkin" wrote:
>>>>
>>>> Craig Markwardt wrote:
>>>> >
>>>> > Well, as grumpy as I have been in the past about IDL
>> wishes, this is
>>>> > one thing I do not want to have in IDL now!
>>>>
>>>> I am with you Craig. Besides, for the purists of array
>> indexing, I think
>>>> it is unfair to dasignate a *lower* array bounds. We don't
>> designate the
>>>> *upper* one.
>>>
>>> In the context of initially declaring an array in IDL, sure you do:
>>>
>>> x = fltarr(10)
>>>
>>> declares the upper bound as 9. We also designate a lower
>> bound: 0. The difference between
>>> the two is that I can change the former.
>>>
> So how about if {flt,dbl,complex,int,lon,dcomplex,byt,str}arr
> and make_array could accept both the form shown above and this form:
>
> y = fltarr(-5:10)
>
> which would declare the lower bound as -5 and the upper bound as 9,
> giving a total of 16 elements, including the zero element? This
> isn't quite as nice as PL/1's method because of the zero element,
> but it would be usable and wouldn't break any existing code. Future
> programs would have to take into consideration that
>
> y = fltarr(-5:-1)
>

> would have a lower bound of -5 and an upper bound of -1, giving
> a total of only 5 elements due to the lack of a zero element. PL/1's
> syntax avoided this problem by having the lower bound default to 1
> if not coded, but I think I could live with it as long as I were
> aware of it.

I guess I was not so concerned with the syntax of indexing, as with the capability of extending lists in two directions without jumping through hoops.

Think Perl. For example, to add an element or elements to the end of a possibly non-existent list, you say:

push @list, $elem;

To pull one element off the end:

pop @list;

To pull one off the front:

shift @list;

To push one onto the front:

unshift @list, $elem;

With such a collection we could rid ourselves of all those silly statements like:

if n_elements(list) eq 0 then list=[elem] else list=[list, elem]

As for actual negative indices, I think that's probably an impossible idea to implement at this point, given how hardcoded the notion of zero offset arrays is in so many processing routines, both built-in and otherwise.

This all harkens back to the notion of zero-length or null vectors, e.g. as a return from where(), and as a possibility for indexing arrays. There were other technical difficulties with implementing this idea, but I find more instances where it would have been useful all the time.

JD

Subject: Re: User selectable lower array bound?
Posted by Craig Markwardt on Tue, 07 Aug 2001 17:21:03 GMT

View Forum Message <> Reply to Message

JD Smith <jdsmith@astro.cornell.edu> writes:
> The most annoying thing about IDL arrays to me is the need always to
> test whether they exist or not when concatenating onto them. The idea
> of extending arrays in both directions would be neatly summed up by
> allowing:
>
> a=[b,a] & a=[a,b] even if a doesn't (yet) exist.
>
> Either that, or IDL needs a list type which allows such operations.
> Wasn't that just me ranting about special case functionality leading to
> inconsistency?

Hmmm, agreed. I think WMC's and my proposal was for a "null" data
type which was essentially an empty list.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: User selectable lower array bound?
Posted by Martin Schultz on Wed, 08 Aug 2001 11:46:32 GMT
View Forum Message <> Reply to Message

Marc Schellens <m_schellens@hotmail.com> writes:

> Ahem:
>
> pro push,list,elem
> if n_elements(list) eq 0 then list= else list=
> end
>
> cheers,
> marc

Won't work if elem is an array ;-(Which dimension do you append?
And if elem is a structure you will soon need something like
Relax_StructAssign...

Martin

--
[[[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[
[[phone: +49 40 41173-308   [[
[[fax: +49 40 41173-298   [[
[[martin.schultz@dkrz.de   [[
[[[[[[[[[[[

Subject: Re: User selectable lower array bound?
Posted by Martin Schultz on Wed, 08 Aug 2001 11:47:53 GMT
View Forum Message <> Reply to Message

Craig Markwardt <craigmnet@cow.physics.wisc.edu> writes:
> JD Smith <jdsmith@astro.cornell.edu> writes:
>> The most annoying thing about IDL arrays to me is the need always to
>> test whether they exist or not when concatenating onto them. The idea
>> of extending arrays in both directions would be neatly summed up by
>> allowing:
>>
>> a= & a= even if a doesn't (yet) exist.
>>
>> Either that, or IDL needs a list type which allows such operations.
>> Wasn't that just me ranting about special case functionality leading to
>> inconsistency?
>
> Hmmm, agreed. I think WMC's and my proposal was for a "null" data
> type which was essentially an empty list.
>
> Craig

as in
  var = nularr(100) ;-)

what is N_Elements(nul_variable) ? How do you test for a keyword etc.?

Cheers,

Martin

--
[[[[[[[[[[[
[[Dr. Martin Schultz   Max-Planck-Institut fuer Meteorologie   [[
[[Bundesstr. 55, 20146 Hamburg   [[

[[phone: +49 40 41173-308 [[
[[fax: +49 40 41173-298 [[
[[martin.schultz@dkrz.de [[
[[[[[[[[[[

Subject: Re: User selectable lower array bound?
Posted by marc schellens[1] on Wed, 08 Aug 2001 14:29:37 GMT
View Forum Message <> Reply to Message

Martin Schultz wrote:
>
> Marc Schellens <m_schellens@hotmail.com> writes:
>
>> Ahem:
>>
>> pro push,list,elem
>> if n_elements(list) eq 0 then list=[elem] else list=[list, elem]
>> end
>>
>> cheers,
>> marc
>
> Won't work if elem is an array ;-(Which dimension do you append?
> And if elem is a structure you will soon need something like
> Relax_StructAssign...
>
> Martin

As Paul pointed already out:
For your demands, write a more sophisticated routine then.
Anyway as already stated:
Everything could be so nice if c=[a,b] with noexistent a or b would
work...

marc

Subject: Re: User selectable lower array bound?
Posted by Paul van Delst on Wed, 08 Aug 2001 15:04:57 GMT
View Forum Message <> Reply to Message

Martin Schultz wrote:
>
> Marc Schellens <m_schellens@hotmail.com> writes:
>
>> Ahem:

>>
>> pro push,list,elem
>> if n_elements(list) eq 0 then list=[elem] else list=[list, elem]
>> end
>>
>> cheers,
>> marc
>
> Won't work if elem is an array ;-(

For rank-1 arrays, shur it will (Can lists, as defined by JD's Perl description be other than rank-1?) I do it all the time. Except that I would do:

if n_elements(list) eq 0 then list=elem else list=[list, elem]

For 2-D arrays you would have to check for conformability in your own code but that's also easy to do. 4 or 5-D arrays would be a problem because at some point IDL chucks a nervous wobbly when you try to do stuff like:

list = [[[[list]]],[[[elem]]]]

or somesuch sort of thing (i.e. too many "[]"'s). I think Craig Markwardt posted something about this once.

> And if elem is a structure you will soon need something like
> Relax_StructAssign...

Shur, but I thought the whole point of something like Marc's push procedure was to eliminate all the nasty details from the user (be they 2-d array or structure details), i.e. it doesn the conformability check or the relaxed structure assignment?

paulv

--
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.

Subject: Re: User selectable lower array bound?
Posted by John-David T. Smith on Wed, 08 Aug 2001 19:26:17 GMT
View Forum Message <> Reply to Message

Marc Schellens wrote:
>
>> Think Perl. For example, to add an element or elements to the end of a

>> possibly non-existent list, you say:
>>
>> push @list, $elem;
>>
>> To pull one element off the end:
>>
>> pop @list;
>>
>> To pull one off the front:
>>
>> shift @list;
>>
>> To push one onto the front:
>>
>> unshift @list, $elem;
>>
>> With such a collection we could rid ourselves of all those silly
>> statements like:
>>
>> if n_elements(list) eq 0 then list=[elem] else list=[list, elem]
>
> Ahem:
>
> pro push,list,elem
> if n_elements(list) eq 0 then list=[elem] else list=[list, elem]
> end

I had one just like this in my collection, but I just didn't feel right spawning a function call and path search just to add an element to a list. A built-in method would be much preferrable.

JD

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.

Subject: Re: User selectable lower array bound?
Posted by david[2] on Thu, 09 Aug 2001 13:51:45 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> 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.

Better to put this at the *top* of the rant,
if you want the good folks at RSI to pay
attention. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: User selectable lower array bound?
Posted by Paul van Delst on Thu, 09 Aug 2001 14:04:27 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Paul van Delst writes:
>
>> 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.
>
> Better to put this at the *top* of the rant,
> if you want the good folks at RSI to pay
> attention. :-)

You're right of course. I guess I'm a serial guy stuck in a parallel world...

. <- that dot is the violin playing for my misfortune :o)

paulv

--
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.
