Subject: Re: Another simple one
Posted by Brian Larsen on Sat, 28 Jul 2007 12:45:28 GMT
View Forum Message <> Reply to Message

Arrays do have to be rectangular, you can't have them with different sizes in different dimensions. That being said you can populate a structure with different 1-d arrays of different lengths or something. The easiest way is probably not to do that however but to keep an array with the indicies you are interested in from the data array. Think where() and give that a try then you can reference it like data[inds] to get the ones you want.

---------- ----------
Brian Larsen
Boston University
Center for Space Physics

Subject: Re: Another simple one
Posted by Carsten Lechte on Sat, 28 Jul 2007 13:19:35 GMT
View Forum Message <> Reply to Message

Brian Larsen wrote:
> Arrays do have to be rectangular, you can't have them with different
> sizes in different dimensions. That being said you can populate a
> structure with different 1-d arrays of different lengths or
> something.

Or, you can use an array of pointers to differently-sized 1D arrays. Then, you can use normal indexing for access, instead of a mix of structure and array access. Chapter 8 of "Building IDL Applications" deals with the concept of pointers.

chl

Subject: Re: Another simple one
Posted by Brian Larsen on Sat, 28 Jul 2007 14:14:01 GMT
View Forum Message <> Reply to Message

> Or, you can use an array of pointers to differently-sized 1D arrays.
> Then, you can use normal indexing for access, instead of a mix of
> structure and array access. Chapter 8 of "Building IDL Applications"
> deals with the concept of pointers.
>
> chl

True, I normally take the "this is IDL not C, I don't have to fight with pointers" route to these types of things, even if the pointer is the better way.

Brian

---------- ----------

Brian Larsen
Boston University
Center for Space Physics

Subject: Re: Another simple one
Posted by MarioIncandenza on Sun, 29 Jul 2007 19:39:23 GMT
View Forum Message <> Reply to Message

On Jul 28, 12:58 am, snudge42 <snudg...@gmail.com> wrote:
> How do I truncate a multidimensional array at a different place for
> each dimension i.e. I start with 3x6 array of values A=
>
> 0 0 0
> 0 0 0
> 0 0 0
> 0 0 0
> 0 0 0
> 0 0 0
>
> and want to truncate each dimension at a fixed value stored in an
> array B, where the values are (2,3,4) for example so that I get:
>
> 0 0 0
> 0 0 0
> 0 0 0
> 0 0
> 0
>

By 'truncate', you mean either "perform calculations on only part of array A" or "write output of only part of array A". For the second case, the I/O penalty is far greater than the for-loop penalty, just use a loop. For the first case, consider this:
;NOTE: this example truncates along rows, you'll need to TRANSPOSE to do columns
btrunc=a*0; initialize truncation helper array

for i=0l,n_elements(b)-1 do btrunc[0]=(lindgen(b[i]+1))+1 gt 0; anyone
care to try getting rid of this loop?
print,trunc
> 1 1 1 0 0
> 1 1 1 1 0
> 1 1 1 1 1
; from here, you can manipulate A in several ways:
; such as setting truncated components to NaN:
ftrunc=where(trunc eq 0)
anew=a & anew[ftrunc]=!values.NaN
; setting truncated components to 0:
anew= a * btrunc;

Hope this helps,

--Edward H.

Subject: Re: Another simple one
Posted by Jean H. on Mon, 30 Jul 2007 16:19:51 GMT
View Forum Message <> Reply to Message

> Is that possible? Or does my array have to be square so have to
> truncate the whole thing at element 4 for example?
>
> Cheers,
>
> Snudge42
>

Hi,

You can also use a 1D array containing all of your data... then you
should know which entries correspond to which line...

ex: a = [1,2,3,4,5,6]

you can think of A as
1,2,3
,4,5
, ,6

So, but this starts to be useful with big arrays, you can have a 2D
array that indexes the 1D array...

ex:
indices2Dto1D = [[0,1,2],[-1,3,4],[-1,-1,5]]
print, "value of 2,2 = ", a[indices2Dto1D[2,2]] ==> 6

and
indices1Dto2D = [0,1,2,4,5,8]
 print, "2D coords of the value 6 = ", indices1Dto2D[where(a eq 6)] ==>
8   (this is the 1D coordinate in the 2D array... you can transform it
back to 2,2)

Jean
PS: I use this all the time to keep satellite images covering a study
area having a crazy shape... I save about 75% of the otherwise required
memory! ... I have to keep only 1 array covering the entire area, and
all the other arrays cover only the study area!

Subject: Re: Another simple one
Posted by snudge42 on Thu, 09 Aug 2007 04:43:48 GMT
View Forum Message <> Reply to Message

On Jul 31, 2:19 am, "Jean H." <jghas...@DELTHIS.ucalgary.ANDTHIS.ca>
wrote:
>> Is that possible? Or does my array have to be square so have to
>> truncate the whole thing at element 4 for example?
>
>> Cheers,
>
>> Snudge42
>
> Hi,
>
> You can also use a 1D array containing all of your data... then you
> should know which entries correspond to which line...
>
> ex: a = [1,2,3,4,5,6]
>
> you can think of A as
> 1,2,3
>   ,4,5
>   , ,6
>
> So, but this starts to be useful with big arrays, you can have a 2D
> array that indexes the 1D array...
>
> ex:
> indices2Dto1D = [[0,1,2],[-1,3,4],[-1,-1,5]]
>   print, "value of 2,2 = ", a[indices2Dto1D[2,2]] ==> 6
> and
> indices1Dto2D = [0,1,2,4,5,8]
>   print, "2D coords of the value 6 = ", indices1Dto2D[where(a eq 6)] ==>
> 8   (this is the 1D coordinate in the 2D array... you can transform it

> back to 2,2)
>
> Jean
> PS: I use this all the time to keep satellite images covering a study
> area having a crazy shape... I save about 75% of the otherwise required
> memory! ... I have to keep only 1 array covering the entire area, and
> all the other arrays cover only the study area!

Lots to think about there, thanks everyone. =)
