Subject: Re: Loop Arrays
Posted by Ken Mankoff on Tue, 16 Oct 2001 17:49:45 GMT
View Forum Message <> Reply to Message

Hi Martin,

Yes, the code you supplied come the closest to doing what I want. Thank you.

I believe I will be honing my perl skills a bit to play with an implementation of this idea. I will let you know how it goes...

-k.

On Mon, 15 Oct 2001, Martin Downing wrote:

>
>
> "Ken Mankoff" <mankoff@I.HATE.SPAM.cs.colorado.edu> wrote in message
> news:Pine.LNX.4.33.0110091423020.29204-100000@snoe.colorado. edu...
>> On Tue, 9 Oct 2001, Mark Hadfield wrote:
>>
>>> From: "Ken Mankoff" <mankoff@lasp.colorado.edu>
>>>> I am interested in creating circular arrays, where subscripts that
> would
>>>> be out-of-bounds on a regular array just start indexing on the other
> side
>>>> of the array.
>>>
>>> You can do quite a lot with ordinary arrays using arrays of indices, eg
>>>
>>> a = indgen(10)
>>> print, a[[0,10,20,100] mod n_elements(a)]
>>>
>>
>> This is the technique I have been using. However there are 2 cases it does
>> not cover:
>>
>> 1) negative indexes require a few more lines of code to get your example
>> to work. I would recode it as:
>>
>> a = indgen(10)
>> indexes = [0,10,20,100,-10,-22] ;;; or some other values...
>> ind = indexes mod n_elements(a)
>> neg = where(ind lt 0, num)
>> if (num ne 0) then ind[neg] = ind[neg] + n_elements(a)
>> print, a[ind]
>>

>> 2) subscript ranges. You cannot do:
>> print, a[8:12 mod n_elements(a)]
>>
>> It is these two specific abilities that I would like to have.
>>
>> -k.
>
> Hi Ken,
>
> This discussion makes for interesting reading. However, except for arrays
> representing objects with circular indexing logic, such as closed
> polygons for instance, I'm not sure it is productive to prevent IDL from
> pointing out that you have run off the end of an array!
>
> Anyway, there is a way you can code range indexing above for circular
> arrays:
>
> eg for indexing a[b:c] do the following:
>
> IDL> a = indgen(10) ; to be interpreted as a circular array
> IDL> b = 9 & c = 13
> IDL> print, a[(indgen(c-b)+b) MOD n_elements(a)]
> ; read as a[b:c]
> 9 0 1 2
>
> IDL> b = 9 & c = 23
> IDL> print, a[(indgen(c-b)+b) MOD n_elements(a)] ; read as a_circ[b:c]
> 9 0 1 2 3 4 5 6 7 8 9 0 1 2
>
>
> -Is that of any use to you?
>
> regards
>
> Martin
>