Subject: Re: Loop Arrays
Posted by David Fanning on Tue, 09 Oct 2001 19:57:25 GMT
View Forum Message <> Reply to Message

Ken Mankoff (mankoff@lasp.colorado.edu) writes:

> 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.
>
> ex:
> a = circleIndgen(10)
> print, a[-1]
> 9
> print, a[11]
> 1
> print, a[[0,10,20,100]]
> 0, 0, 0, 0
>
> print, a[8:11]
> 8, 9, 0, 1
>
> ;;; not sure if this makes sense, but i think it can easily be
> ;;; done if the rest is possible...
> print, a[8:2]
> 8, 9, 0, 1
>
> I think that overloading the [] operators is not an option from my
> understanding of IDL. Does anyone know if this is possible?

Uh, you must have dropped your notes from your C++ course
and got them mixed up with your print-outs of IDL
newsgroup articles. :-)

There isn't going to be any "overloading of operators"
in IDL, I can assure you of that.

Although you *could* create an object that could produce
the results you want when you call, for example, the
SubSet method (or whatever).

> Desirable? Dumb?

Depends entirely on what you are trying to do.
Although it does seem like an awful lot of work
to me. :-)

Cheers,

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

Subject: Re: Loop Arrays
Posted by Mark Hadfield on Tue, 09 Oct 2001 19:59:39 GMT
View Forum Message <> Reply to Message

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.

> I think that overloading the [] operators is not an option from my
> understanding of IDL. Does anyone know if this is possible?

It's not possible. IDL does not support operator overloading.

> Desirable? Dumb?

Maybe. No.

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)]
```

BTW another indexing extension that appeals to me is Python's use of
negative indices to refer to positions relative to the end of the array,
eg -1 refers to the rightmost element.

Mark Hadfield
m.hadfield@niwa.cri.nz http://katipo.niwa.cri.nz/~hadfield
National Institute for Water and Atmospheric Research

--
Posted from clam.niwa.cri.nz [202.36.29.1]
via Mailgate.ORG Server - http://www.Mailgate.ORG

Subject: Re: Loop Arrays
Posted by Ken Mankoff on Tue, 09 Oct 2001 22:06:36 GMT
View Forum Message <> Reply to Message

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.

--
Ken Mankoff
LASP://303.492.3264
http://lasp.colorado.edu/~mankoff/

Subject: Re: Loop Arrays
Posted by Ken Mankoff on Mon, 15 Oct 2001 16:35:09 GMT
View Forum Message <> Reply to Message

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.
> BTW another indexing extension that appeals to me is Python's use of
> negative indices to refer to positions relative to the end of the array,
> eg -1 refers to the rightmost element.

I have a few ideas for implementing this feature. If you are interested in the option and ability to do this, please let me know your suggestions on the following.

The algorithm is simple: Evaluate the numbers, variables, and functions that make up the subscripts w.r.t. the dimensions of the arrays. If any are negative or larger than their dimension, then re-write code to behave in new way.

What is needed is a regular expression filter and replacement algorithm. If any of the following is detected, then replace with:

```
  a[ -3 ] a[ n_elements(a)-3 ]
  a[ 3-n ] if( 3-n lt 0 ) then a[ n_elements(a)-3-n ] else a[3-n]
  a[ 2, -3 ] a[ 2, (size(a,/dim))[2] -3 ]
  a[ 2:-4 ] a[ 2: n_elements(a)-4 ]
  a( -3 ) * let normal interpreter handle ()
```

etc...

I wrote this in IDL (for about 1/2 the cases) before I realized it has some pitfalls: retall and .full_reset_session exit the routine, and this routine tries to emulate $MAIN$, which is bad.

I realized I can fix those problemn with emacs lisp or Perl. With either one, I can open a pipe between IDL and the user, give them the ability turn on/off wierd_subscript_method, and then reformat their input so the IDL interpreter understands it. This could be done in any language, but I assume that Perl would be best since it is good with regexp and string manipulations. Does anyone have any other suggestions?

-k.

Subject: Re: Loop Arrays
Posted by Martin Downing on Mon, 15 Oct 2001 21:43:51 GMT
View Forum Message <> Reply to Message

"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

> --
> Ken Mankoff
> LASP://303.492.3264
> http://lasp.colorado.edu/~mankoff/