Subject: Re: IDL FOR Loop variable increments
Posted by David Fanning on Thu, 18 Sep 2008 20:03:30 GMT
View Forum Message <> Reply to Message

Raghu writes:

> In the code above, i want b (the number of bands) to be incremented
> by the value (b+k) instead of standard consistent increments of 1 or
> 2.. So, instead of b progressing from 0 to 8 as 0,1,2,3 etc..i want it
> to iterate based on the value of (b+k) derived out of the inner FOR
> loop (for k=1, j). My results run correctly but those pixels for which
> 'b' needs a (b+k) increment are not recogized by the line "if b gt 0
> and b ne (c) then continue" and so it keeps skipping all 'b'
> iterations. 'c' here is just a variable assigned to the value of b+k.
>
> Where am i going wrong ?

I think you have clearly chosen the wrong program
construction. You should NEVER be changing a loop
variable inside the loop programmatically. (I don't
think IDL 7 even allows it, since I was inadvertently
doing it the other day and got my hand slapped.) Maybe
you want a WHILE loop.

Cheers,

David

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: IDL FOR Loop variable increments
Posted by pgrigis on Thu, 18 Sep 2008 20:23:07 GMT
View Forum Message <> Reply to Message

Raghu wrote:
> Hi all,
>
> I'm having a problem with FOR loop increments in IDL which i'm not
> able to fix. I have pasted the code and a small description about it
> follows.
> k=1
> for b=0,8 do begin ; Number of bands
> j=7
> c=b+k

k here is going to be equal 1 for the first iteration, and equal j
afterwards

Paolo

> if b gt 0 and b ne (c) then continue
> for k=1, j do begin ; Iterations within bands
> if c lt 7 then begin
> if finite(ndvislice[s,b+k]) eq 1 then begin
> if ndvislice[s,b+k] ge ndvislice[s,b] then begin
> ndvi[s,b+k]=ndvislice[s,b+k]
> break
> endif else begin
> if ndvislice[s,b+k] lt ndvislice[s,b] then begin
> ndvi[s,b+k]=mask[s,r]
> endif
> endelse
> endif else begin
> ndvi[s,b+k]=mask[s,r]
> endelse
> endif
> endfor
> ENDFOR
>
> In the code above, i want b (the number of bands) to be incremented
> by the value (b+k) instead of standard consistent increments of 1 or
> 2.. So, instead of b progressing from 0 to 8 as 0,1,2,3 etc..i want it
> to iterate based on the value of (b+k) derived out of the inner FOR
> loop (for k=1, j). My results run correctly but those pixels for which
> 'b' needs a (b+k) increment are not recogized by the line "if b gt 0
> and b ne (c) then continue" and so it keeps skipping all 'b'
> iterations. 'c' here is just a variable assigned to the value of b+k.
>
> Where am i going wrong ? Please let me know if you need more
> information.
>
> Thanks,
> Raghu

Subject: Re: IDL FOR Loop variable increments
Posted by Jean H. on Thu, 18 Sep 2008 21:52:23 GMT
View Forum Message <> Reply to Message

> I think you have clearly chosen the wrong program
> construction. You should NEVER be changing a loop
> variable inside the loop programmatically. (I don't
> think IDL 7 even allows it, since I was inadvertently

> doing it the other day and got my hand slapped.) Maybe
> you want a WHILE loop.

No problem to do that here..
{ x86 Win32 Windows Microsoft Windows 7.0.1 Mar 20 2008 32 64}

```
pro del
  for i=0,10 do begin
    if i eq 5 then i = 8
    print,i
  endfor
end

==>
       0
       1
       2
       3
       4
       8
       9
      10
```

Could you comment on the "risk" of changing the loop counter within the loop?

Jean

Subject: Re: IDL FOR Loop variable increments
Posted by Jean H. on Thu, 18 Sep 2008 22:00:56 GMT
View Forum Message <> Reply to Message

> k=1
> for b=0,8 do begin ; Number of bands
> j=7
> c=b+k
> if b gt 0 and b ne (c) then continue

So, in this case, you go back to the for loop and increase b...
so, you have the first iteration with b=0 (this above statement is not executed when b=0). Then, you increase b, so 'b gt 0' is true, and 'b ne c' is also always true... k >= 1, so b+ k will NEVER be equal to b ...
so the above statement is always executed, until b > 8.
Jean

> for k=1, j do begin ; Iterations within bands
> if c lt 7 then begin
> if finite(ndvislice[s,b+k]) eq 1 then begin
> if ndvislice[s,b+k] ge ndvislice[s,b] then begin
> ndvi[s,b+k]=ndvislice[s,b+k]
> break
> endif else begin
> if ndvislice[s,b+k] lt ndvislice[s,b] then begin
> ndvi[s,b+k]=mask[s,r]
> endif
> endelse
> endif else begin
> ndvi[s,b+k]=mask[s,r]
> endelse
> endif
> endfor
> ENDFOR
>
> In the code above, i want b (the number of bands) to be incremented
> by the value (b+k) instead of standard consistent increments of 1 or
> 2.. So, instead of b progressing from 0 to 8 as 0,1,2,3 etc..i want it
> to iterate based on the value of (b+k) derived out of the inner FOR
> loop (for k=1, j). My results run correctly but those pixels for which
> 'b' needs a (b+k) increment are not recogized by the line "if b gt 0
> and b ne (c) then continue" and so it keeps skipping all 'b'
> iterations. 'c' here is just a variable assigned to the value of b+k.
>
> Where am i going wrong ? Please let me know if you need more
> information.
>
> Thanks,
> Raghu

Subject: Re: IDL FOR Loop variable increments
Posted by David Fanning on Thu, 18 Sep 2008 22:46:14 GMT
View Forum Message <> Reply to Message

Jean H writes:

>> I think you have clearly chosen the wrong program
>> construction. You should NEVER be changing a loop
>> variable inside the loop programmatically. (I don't
>> think IDL 7 even allows it, since I was inadvertently
>> doing it the other day and got my hand slapped.) Maybe
>> you want a WHILE loop.
>
> No problem to do that here..

> { x86 Win32 Windows Microsoft Windows 7.0.1 Mar 20 2008 32 64}

Humm. I was afraid of that. I was too much in the middle
of something to stop and see what the real problem was.
I was passing a loop variable into another program as a
parameter, and that other program was, on occasion, changing
the passed-in variable to something else, based on some
programming logic. I was getting some kind of error about
"cannot change this kind of variable".

I don't know. I've never seen the error before, and I can't
reproduce it now in a quick test program, so... Probably
didn't happen. :-)

> Could you comment on the "risk" of changing the loop counter within the
> loop?

I was in a hurry and late for a meeting when I wrote this,
but as I was going on the door, I realized that of course
we often change "loop variables" inside of loops. In fact,
the loop I suggested, a WHILE loop, nearly *always* does that.
So, nonsense.

The point I was trying to make, though, is that in
a counting loop, like a FOR loop, it seems to me that
changing the counter in the program is a terrible idea.
I just can't think of any time it is a good idea. I think
it is a programming style that just flat out leads to
programming errors. If you think you have to do it, then
I think you ought to re-think what you are doing, and choose
another programming structure that lends itself more easily
to the task.

I've just never seen anything good come of it.

Cheers,

David

P.S. Let's just say that since I now have to work for a living, I
find there is a LOT less time to actually think about what I am
writing. I am MUCH more sympathetic to the kind of advice that
is typical on any newsgroup except this one. :-)

--
David Fanning, Ph.D.
Coyote's Guide to IDL Programming (www.dfanning.com)

Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: IDL FOR Loop variable increments
Posted by raghuram on Fri, 19 Sep 2008 01:31:44 GMT
View Forum Message <> Reply to Message

On Sep 18, 2:52 pm, Jean H <jghas...@DELTHIS.ucalgary.ANDTHIS.ca>
wrote:

> Could you comment on the "risk" of changing the loop counter within the
> loop?
>
> Jean

Ok here's the scene. In my example, lets say b=0 and k=2 (i.e. b+k=
0+2 = 2) satisfies my condition, thats like saying i'm comparing
Bands '0' and '2' as in line 9, "if ndvislice[s,b+k] ge ndvislice[s,b]
then begin". The next thing i want to do is to
compare bands '2' and '3'. Its because band 1 does not satisy my
condition and i don't want the code to look at band 1 again.
Hence i want to skip from band 0 to band 2. This is primarily because
when k was 1 (b+k=0+1=1), the condition was not satisfied which
is why 'k' changes to 2. The increments are not necessarily consistent
either. I may want to skip from band 4 to band 7 if bands 5 and 6
do not satisfy my condition which in the above code is on the 9th line
"if ndvislice[s,b+k] ge ndvislice[s,b] then begin ".
So, the reason why i try to set b to b+k is just so that 'b' turns to
'b+k' as in, b=0 to 'b+2" which means i want the new b,
or the b in the next FOR loop iteration to change to 2 from 0 instead
of changing to 1.
Maybe there's a better way to do this. I hope i'm being clear.

Looking forward to a reply,
Raghu

Subject: Re: IDL FOR Loop variable increments
Posted by Carsten Lechte on Fri, 19 Sep 2008 08:07:34 GMT
View Forum Message <> Reply to Message

Jean H wrote:
> Could you comment on the "risk" of changing the loop counter within the
> loop?

(You did not ask me, but I will comment anyway;)

The risk is called "obfuscation". Outside of certain competitions,
obfuscation is usually considered a thing to avoid. The first statement
in a FOR loop is a nice big billboard that clearly states: "This variable i
will consecutively hold the values from 0 to 10." Imagine your surprise
when i only iterates over some of those values, or even over some totally
different ones when you set i = -3 inside the loop body. Unless you also attach
a blinking comment that notes this unexpected behaviour, future generations
(including yourself in three months) will curse your name when they try to make
sense of your code.

Another thing to consider is if the changing of the loop variable results
in undefined behaviour. IDL seems to follow C's "anything goes" philosophy,
so it will probably work reliably. However, you are still forbidden to
change the type of the loop variable inside the loop;)

chl

Subject: Re: IDL FOR Loop variable increments
Posted by Carsten Lechte on Fri, 19 Sep 2008 08:09:11 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> programming logic. I was getting some kind of error about
> "cannot change this kind of variable".

Maybe the program tried to change the type of the variable?
That would generate an error.

chl

Subject: Re: IDL FOR Loop variable increments
Posted by David Fanning on Fri, 19 Sep 2008 13:13:19 GMT
View Forum Message <> Reply to Message

Carsten Lechte writes:

> Maybe the program tried to change the type of the variable?
> That would generate an error.

Yes, that was the error. I was using a less-than operator
to compare the loop variable (an integer) with a structure
variable, which was a long. This had the effect of changing
the type of the loop variable.

Thanks for the hint.

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: IDL FOR Loop variable increments
Posted by pgrigis on Fri, 19 Sep 2008 14:26:29 GMT
View Forum Message <> Reply to Message

It's nice sometimes to see an IDL error message I've never seen
before!

For the record:
IDL> FOR i =0,10 DO i=long(i)
% Type of FOR statement index variable I may not be changed.
% Execution halted at: $MAIN$ 1 /tmp/idltemp45235Sz

Ciao,
Paolo

David Fanning wrote:
> Carsten Lechte writes:
>
>> Maybe the program tried to change the type of the variable?
>> That would generate an error.
>
> Yes, that was the error. I was using a less-than operator
> to compare the loop variable (an integer) with a structure
> variable, which was a long. This had the effect of changing
> the type of the loop variable.
>
> Thanks for the hint.
>
> Cheers,
>
> David
>
> --
> David Fanning, Ph.D.

> Fanning Software Consulting, Inc.
> Coyote's Guide to IDL Programming: http://www.dfanning.com/
> Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: IDL FOR Loop variable increments
Posted by MarioIncandenza on Fri, 19 Sep 2008 16:53:17 GMT
View Forum Message <> Reply to Message

Raghu,

Not knowing the makeup of NDVI, NDVISLICE, and MASK in your example, it's very difficult to determine what you're up to here, but it strongly looks to me like some higher-order functions are in order here. You are aware that IDL can give you an array MAX(VAR,DIM=N) or MIN, TOTAL, or can do any boolean comparison you choose across an entire array?

As I said, I don't really know what the outcome of your code should be, but even in this economy, I'll bet $5 you can do it with a much smaller, more comprehensible set of array operations.

--Edward H.

Subject: Re: IDL FOR Loop variable increments
Posted by R.G. Stockwell on Fri, 19 Sep 2008 18:53:34 GMT
View Forum Message <> Reply to Message

"Jean H" <jghasban@DELTHIS.ucalgary.ANDTHIS.ca> wrote in message news:gauiil$u32$1@news.ucalgary.ca...
...
> Could you comment on the "risk" of changing the loop counter within the
> loop?

my 2 cents.

First, it is in changing the counter of a for loop.
A for loop explicitly outlines what all counter variables will be.

There are two things:

1) infinite loop, one could easily change the counter to never reach the end condition. A (valid) for loop will always reach the end condition.

2) more insidious, you could inadvertantly cast the counter to a float from

an int, and then have one extra (and unintended) statement executed.

instead of 0,1,2,3,4,5,6 (and not executing i = 7) you could get
0,1,2,3,4,4.99999999,5.99999,6.99999999, (and effectively executing the
extra i ~ 7 step).

Cheers,
bob

Subject: Re: IDL FOR Loop variable increments
Posted by raghuram on Fri, 19 Sep 2008 19:07:54 GMT
View Forum Message <> Reply to Message

On Sep 19, 9:53 am, Ed Hyer <ejh...@gmail.com> wrote:
> Raghu,
>
> Not knowing the makeup of NDVI, NDVISLICE, and MASK in your example,
> it's very difficult to determine what you're up to here, but it
> strongly looks to me like some higher-order functions are in order
> here. You are aware that IDL can give you an array MAX(VAR,DIM=N) or
> MIN, TOTAL, or can do any boolean comparison you choose across an
> entire array?
>
> As I said, I don't really know what the outcome of your code should
> be, but even in this economy, I'll bet $5 you can do it with a much
> smaller, more comprehensible set of array operations.
>
> --Edward H.

You are right. I just haven't thought about using array operations
probably because
a working using these loops doesn't take very long to run. From what i
read here,
it looks a FOR loop may not be the best option for such operations.
Maybe i'll
use a while loop and see what happens or i'll rethink how the code can
be
written in a much more efficient manner. I do know that IDL can do
things like
MAx, TOtal etc. But in my case, all i'm trying to do is to assign
values from
one array(ndvislice) to another(ndvi) based on ndvislice's previious
values.
The outcome i'd expect for each output would be a finite value or a
NaN value.
So i assign my output(ndvi) to ndvislice if conditions are met, else
i'll assign

a NaN value(Mask).

Raghu

Subject: Re: IDL FOR Loop variable increments
Posted by pgrigis on Fri, 19 Sep 2008 19:09:25 GMT
View Forum Message <> Reply to Message

R.G. Stockwell wrote:
> "Jean H" <jghasban@DELTHIS.ucalgary.ANDTHIS.ca> wrote in message
> news:gauiil$u32$1@news.ucalgary.ca...
> ...
>> Could you comment on the "risk" of changing the loop counter within the
>> loop?
>
> my 2 cents.
>
> First, it is in changing the counter of a for loop.
> A for loop explicitly outlines what all counter variables will be.
>
> There are two things:
>
> 1) infinite loop, one could easily change the counter to never
> reach the end condition. A (valid) for loop will always reach the end
> condition.
>
> 2) more insidious, you could inadvertantly cast the counter to a float from
> an int, and then have one extra (and unintended) statement executed.
This seems not to be possible in IDL, as loop counters, unlike normal
variables, cannot change their type.

Ciao,
Paolo

>
> instead of 0,1,2,3,4,5,6 (and not executing i = 7) you could get
> 0,1,2,3,4,4.99999999,5.99999,6.99999999, (and effectively executing the
> extra i ~ 7 step).
>
> Cheers,
> bob

Subject: Re: IDL FOR Loop variable increments
Posted by raghuram on Sat, 20 Sep 2008 05:51:42 GMT

On Sep 19, 12:09 pm, pgri...@gmail.com wrote:
> R.G. Stockwell wrote:
>> "Jean H" <jghas...@DELTHIS.ucalgary.ANDTHIS.ca> wrote in message
>> news:gauiil$u32$1@news.ucalgary.ca...
>> ...
>>> Could you comment on the "risk" of changing the loop counter within the
>>> loop?
>
>> my 2 cents.
>
>> First, it is in changing the counter of a for loop.
>> A for loop explicitly outlines what all counter variables will be.
>
>> There are two things:
>
>> 1) infinite loop, one could easily change the counter to never
>> reach the end condition. A (valid) for loop will always reach the end
>> condition.
>
>> 2) more insidious, you could inadvertantly cast the counter to a float from
>> an int, and then have one extra (and unintended) statement executed.
>
> This seems not to be possible in IDL, as loop counters, unlike normal
> variables, cannot change their type.
>
> Ciao,
> Paolo
>
>
>
>> instead of 0,1,2,3,4,5,6 (and not executing i = 7) you could get
>> 0,1,2,3,4,4.99999999,5.99999,6.99999999, (and effectively executing the
>> extra i ~ 7 step).
>
>> Cheers,
>> bob

Hi all,

Thanks for your replies. Just as David mentioned in his first
response, a while loop worked out much better. Within a single while
loop, i was able to accomplish the task, albeit a bit slowly because
of the non-array operation.

Thanks !

Subject: Re: IDL FOR Loop variable increments
Posted by Wasit.Weather on Sun, 21 Sep 2008 13:44:02 GMT
View Forum Message <> Reply to Message

On Sep 20, 12:51 am, Raghu <raghuram.narasim...@gmail.com> wrote:
> On Sep 19, 12:09 pm, pgri...@gmail.com wrote:
>
>
>
>
>
>
>> R.G. Stockwell wrote:
>>> "Jean H" <jghas...@DELTHIS.ucalgary.ANDTHIS.ca> wrote in message
>>> news:gauiil$u32$1@news.ucalgary.ca...
>>> ...
>>>> Could you comment on the "risk" of changing the loop counter within the
>>>> loop?
>
>>> my 2 cents.
>
>>> First, it is in changing the counter of a for loop.
>>> A for loop explicitly outlines what all counter variables will be.
>
>>> There are two things:
>
>>> 1) infinite loop, one could easily change the counter to never
>>> reach the end condition. A (valid) for loop will always reach the end
>>> condition.
>
>>> 2) more insidious, you could inadvertantly cast the counter to a float from
>>> an int, and then have one extra (and unintended) statement executed.
>
>> This seems not to be possible in IDL, as loop counters, unlike normal
>> variables, cannot change their type.
>
>> Ciao,
>> Paolo
>
>>> instead of 0,1,2,3,4,5,6 (and not executing i = 7) you could get
>>> 0,1,2,3,4,4.99999999,5.99999,6.99999999, (and effectively executing the
>>> extra i ~ 7 step).
>
>>> Cheers,
>>> bob
>
> Hi all,
>
> Thanks for your replies. Just as David mentioned in his first
> response, a while loop worked out much better. Within a single while

> loop, i was able to accomplish the task, albeit a bit slowly because
> of the non-array operation.
>
> Thanks !- Hide quoted text -
>
> - Show quoted text -

Why not you do not share your final results with us to close this
post.
Elkunn

Subject: Re: IDL FOR Loop variable increments
Posted by raghuram on Sun, 21 Sep 2008 22:37:19 GMT
View Forum Message <> Reply to Message

On Sep 21, 6:44 am, Bulrush <Wasit.Weat...@gmail.com> wrote:
> On Sep 20, 12:51 am, Raghu <raghuram.narasim...@gmail.com> wrote:
>
>
>
>> On Sep 19, 12:09 pm, pgri...@gmail.com wrote:
>
>>> R.G. Stockwell wrote:
>>>> "Jean H" <jghas...@DELTHIS.ucalgary.ANDTHIS.ca> wrote in message
>>>> news:gauiil$u32$1@news.ucalgary.ca...
>>>> ...
>>>> > Could you comment on the "risk" of changing the loop counter within the
>>>> > loop?
>
>>>> my 2 cents.
>
>>>> First, it is in changing the counter of a for loop.
>>>> A for loop explicitly outlines what all counter variables will be.
>
>>>> There are two things:
>
>>>> 1) infinite loop, one could easily change the counter to never
>>>> reach the end condition. A (valid) for loop will always reach the end
>>>> condition.
>
>>>> 2) more insidious, you could inadvertantly cast the counter to a float from
>>>> an int, and then have one extra (and unintended) statement executed.
>
>>> This seems not to be possible in IDL, as loop counters, unlike normal
>>> variables, cannot change their type.
>
>>> Ciao,

>>> Paolo
>
>>>> instead of 0,1,2,3,4,5,6 (and not executing i = 7) you could get
>>>> 0,1,2,3,4,4.99999999,5.99999,6.99999999, (and effectively executing the
>>>> extra i ~ 7 step).
>
>>>> Cheers,
>>>> bob
>
>> Hi all,
>
>> Thanks for your replies. Just as David mentioned in his first
>> response, a while loop worked out much better. Within a single while
>> loop, i was able to accomplish the task, albeit a bit slowly because
>> of the non-array operation.
>
>> Thanks !- Hide quoted text -
>
>> - Show quoted text -
>
> Why not you do not share your final results with us to close this
> post.
> Elkunn

Hi,

I will. I don't have the code with me this weekend. I'll post it on
Monday at work.

Thanks,
Raghu

Subject: Re: IDL FOR Loop variable increments
Posted by raghuram on Mon, 22 Sep 2008 15:54:59 GMT
View Forum Message <> Reply to Message

On Sep 21, 3:37 pm, Raghu <raghuram.narasim...@gmail.com> wrote:
> On Sep 21, 6:44 am, Bulrush <Wasit.Weat...@gmail.com> wrote:
>
>
>
>> On Sep 20, 12:51 am, Raghu <raghuram.narasim...@gmail.com> wrote:
>
>>> On Sep 19, 12:09 pm, pgri...@gmail.com wrote:
>
>>>> R.G. Stockwell wrote:
>>>> > "Jean H" <jghas...@DELTHIS.ucalgary.ANDTHIS.ca> wrote in message

>>>> >news:gauiil$u32$1@news.ucalgary.ca...
>>>> > ...
>>>> > > Could you comment on the "risk" of changing the loop counter within the
>>>> > > loop?
>
>>>> > my 2 cents.
>
>>>> > First, it is in changing the counter of a for loop.
>>>> > A for loop explicitly outlines what all counter variables will be.
>
>>>> > There are two things:
>
>>>> > 1) infinite loop, one could easily change the counter to never
>>>> > reach the end condition. A (valid) for loop will always reach the end
>>>> > condition.
>
>>>> > 2) more insidious, you could inadvertantly cast the counter to a float from
>>>> > an int, and then have one extra (and unintended) statement executed.
>
>>>> This seems not to be possible in IDL, as loop counters, unlike normal
>>>> variables, cannot change their type.
>
>>>> Ciao,
>>>> Paolo
>
>>>> > instead of 0,1,2,3,4,5,6 (and not executing i = 7) you could get
>>>> > 0,1,2,3,4,4.99999999,5.99999,6.99999999, (and effectively executing the
>>>> > extra i ~ 7 step).
>
>>>> > Cheers,
>>>> > bob
>
>>> Hi all,
>
>>> Thanks for your replies. Just as David mentioned in his first
>>> response, a while loop worked out much better. Within a single while
>>> loop, i was able to accomplish the task, albeit a bit slowly because
>>> of the non-array operation.
>
>>> Thanks !- Hide quoted text -
>
>>> - Show quoted text -
>
>> Why not you do not share your final results with us to close this
>> post.
>> Elkunn
>
> Hi,

>
> I will. I don't have the code with me this weekend. I'll post it on
> Monday at work.
>
> Thanks,
> Raghu

Hello,

here is the while loop piece of code that works now.

```
b=0
k=1

while (b+k lt nb) do begin

if finite(ndvislice[s,b+k]) eq 0 or finite (ndsislice[s,b+k]) eq 0
then begin
ndvi[s,b+k]=mask[s,r]
ndsi[s,b+k]=mask[s,r]
k=k+1
endif else begin
if (ndvislice[s,b+k] lt ndvislice[s,b])then begin;
ndvi[s,b+k]=mask[s,r]
ndsi[s,b+k]=mask[s,r]
k=k+1
endif else begin
ndvi[s,b+k]=ndvislice[s,b+k]
ndsi[s,b+k]=ndsislice[s,b+k]
b=b+k
k=1
endelse
endelse
endwhile
```

So, with two conditions, i can change the number of bands (b) or i can change the counter (k) depending on which condition is satisfied during an iteration. Although it works, it does take some time which might suggest that an array oriented code might work faster. But anyway, it works.

Thanks!

Raghu

Subject: Re: IDL FOR Loop variable increments
Posted by Jeremy Bailin on Tue, 23 Sep 2008 14:48:46 GMT
View Forum Message <> Reply to Message

On Sep 22, 11:54 am, Raghu <raghuram.narasim...@gmail.com> wrote:
> On Sep 21, 3:37 pm, Raghu <raghuram.narasim...@gmail.com> wrote:
>
>
>
>> On Sep 21, 6:44 am, Bulrush <Wasit.Weat...@gmail.com> wrote:
>
>>> On Sep 20, 12:51 am, Raghu <raghuram.narasim...@gmail.com> wrote:
>
>>>> On Sep 19, 12:09 pm, pgri...@gmail.com wrote:
>
>>>> > R.G. Stockwell wrote:
>>>> > > "Jean H" <jghas...@DELTHIS.ucalgary.ANDTHIS.ca> wrote in message
>>>> > >news:gauiil$u32$1@news.ucalgary.ca...
>>>> > > ...
>>>> > > > Could you comment on the "risk" of changing the loop counter within the
>>>> > > > loop?
>
>>>> > > my 2 cents.
>
>>>> > > First, it is in changing the counter of a for loop.
>>>> > > A for loop explicitly outlines what all counter variables will be.
>
>>>> > > There are two things:
>
>>>> > > 1) infinite loop, one could easily change the counter to never
>>>> > > reach the end condition. A (valid) for loop will always reach the end
>>>> > > condition.
>
>>>> > > 2) more insidious, you could inadvertantly cast the counter to a float from
>>>> > > an int, and then have one extra (and unintended) statement executed.
>
>>>> > This seems not to be possible in IDL, as loop counters, unlike normal
>>>> > variables, cannot change their type.
>
>>>> > Ciao,
>>>> > Paolo
>
>>>> > > instead of 0,1,2,3,4,5,6 (and not executing i = 7) you could get
>>>> > > 0,1,2,3,4,4.99999999,5.99999,6.99999999, (and effectively executing the
>>>> > > extra i ~ 7 step).
>
>>>> > > Cheers,
>>>> > > bob
>

>>>> Hi all,
>
>>>> Thanks for your replies. Just as David mentioned in his first
>>>> response, a while loop worked out much better. Within a single while
>>>> loop, i was able to accomplish the task, albeit a bit slowly because
>>>> of the non-array operation.
>
>>>> Thanks !- Hide quoted text -
>
>>>> - Show quoted text -
>
>>> Why not you do not share your final results with us to close this
>>> post.
>>> Elkunn
>
>> Hi,
>
>> I will. I don't have the code with me this weekend. I'll post it on
>> Monday at work.
>
>> Thanks,
>> Raghu
>
> Hello,
>
> here is the while loop piece of code that works now.
>
> b=0
> k=1
>
> while (b+k lt nb) do begin
>
> if finite(ndvislice[s,b+k]) eq 0 or finite (ndsislice[s,b+k]) eq 0
> then begin
> ndvi[s,b+k]=mask[s,r]
> ndsi[s,b+k]=mask[s,r]
> k=k+1
> endif else begin
> if (ndvislice[s,b+k] lt ndvislice[s,b])then begin;
> ndvi[s,b+k]=mask[s,r]
> ndsi[s,b+k]=mask[s,r]
> k=k+1
> endif else begin
> ndvi[s,b+k]=ndvislice[s,b+k]
> ndsi[s,b+k]=ndsislice[s,b+k]
> b=b+k
> k=1
> endelse

> endelse
> endwhile
>
> So, with two conditions, i can change the number of bands (b) or i can
> change the counter (k) depending on which condition is satisfied
> during an iteration. Although it works, it does take some time which
> might suggest that an array oriented code might work faster. But
> anyway, it works.
>
> Thanks!
>
> Raghu

Okay, I think I finally understand what you're trying to do. Let me
try to encapsulate it in words:

As long as ndvislice is not dropping (with NaNs treated as missing
data), ndvi and ndsi get set to ndvislice and ndsislice respectively,
otherwise they get set to the mask value.
(and ndvi[0] and ndsi[0] are unchanged from the input)

In which case, I think that this is a vectorized version:

```
; mask out all values at first, and only set the relevant ones
ndvi[s,1:*] = mask[s,r]
ndsi[s,1:*] = mask[s,r]
; ignore missing data
wherefinite = where(finite(ndvislice[s,*]) ne 0 and
finite(ndsislice[s,*]) ne 0, nfinite)
if nfinite gt 0 then begin
  ; which ones are greater than or equal to previous good value?
  wherenodrop = where(ndvislice[s,wherefinite[1:*]] ge
ndvislice[s,wherefinite], nnodrop)
  if nnodrop gt 0 then begin
    ; get the index back into the original array
    nodrop_i = wherefinite[wherenodrop]+1
    ndvi[s,nodrop_i] = ndvislice[s,nodrop_i]
    ndsi[s,nodrop_i] = ndsislice[s,nodrop_i]
  endif
endif
```

-Jeremy.