Subject: Avoiding redefinition of variable within loop.
Posted by kcwhite91 on Tue, 10 Jun 2014 15:43:06 GMT
View Forum Message <> Reply to Message

I have a 2-d array, with fixed first dimension. The second dimension changes length with each iteration of the loop through the first dimension. I want to compare each iteration to the one before it to find indices that disappeared between iterations, but because of the varying size I have to redefine it every time. I think there may be a way to avoid this using pointers and a structure, but I'm not sure how to do this.

```
for j=0, 18 do begin & $
l=where(zall[*,*,1,j] ne -32768.0) & $
k=where(zall1[l,1,j] gt 10) & $
index=intarr(19, n_elements(k)) & $
index[j, *]=l[k] & $
if (n_elements(k) gt 1) then begin & $
indv_echotop=fltarr(n_elements(k)) & $
      if (j ne 0) then begin & $
      for b=0, n_elements(index[j,*]-1) do begin & $
      x=where(index[j-1, *] eq index[j, b]) & $
      endfor & $
      endif & $
endif & $
endfor
```

index is my issue. I believe I need to define it as a structure with 19 different fields, one for each iteration.

Any help would be greatly appreciated.

Subject: Re: Avoiding redefinition of variable within loop.
Posted by Matthew Argall on Tue, 10 Jun 2014 17:23:51 GMT
View Forum Message <> Reply to Message

Why not use the COMPLEMENT keyword in the Where() function? If zall and zall1 are the same size, then this would work:

```
keep_indices = where((zall[*,*,1,j] ne -32768.0) and (zall1[*,*,1,j] gt 10),
COMPLEMENT=lost_indices)
```

Subject: Re: Avoiding redefinition of variable within loop.
Posted by kcwhite91 on Tue, 10 Jun 2014 19:05:02 GMT
View Forum Message <> Reply to Message

On Tuesday, June 10, 2014 12:23:51 PM UTC-5, Matthew Argall wrote:

> Why not use the COMPLEMENT keyword in the Where() function? If zall and zall1 are the same size, then this would work:
>
>
>
> keep_indices = where((zall[*,*,1,j] ne -32768.0) and (zall1[*,*,1,j] gt 10), COMPLEMENT=lost_indices)

Sorry, had to reform zall into zall1 because of the first where statement.
zall1=reform(zall, 10201, 4992, 19)

Subject: Re: Avoiding redefinition of variable within loop.
Posted by Matthew Argall on Tue, 10 Jun 2014 20:25:30 GMT
View Forum Message <> Reply to Message

> Sorry, had to reform zall into zall1 because of the first where statement.

Then I think this should still work, unless I misunderstand what you are trying to do...

keep_indices = where((zall[*,*,1,j] ne -32768.0) and (zall[*,*,1,j] gt 10), COMPLEMENT=lost_indices)

Depending on what your purpose is, you might not even need to loop over j. You could just do

keep_indices = where((zall ne -32768.0) and (zall gt 10), COMPLEMENT=lost_indices)
zall[lost_indices] = !values.f_nan

Subject: Re: Avoiding redefinition of variable within loop.
Posted by kcwhite91 on Tue, 10 Jun 2014 21:42:25 GMT
View Forum Message <> Reply to Message

On Tuesday, June 10, 2014 3:25:30 PM UTC-5, Matthew Argall wrote:
>> Sorry, had to reform zall into zall1 because of the first where statement.
>
>
>
> Then I think this should still work, unless I misunderstand what you are trying to do...
>
>
>
> keep_indices = where((zall[*,*,1,j] ne -32768.0) and (zall[*,*,1,j] gt 10), COMPLEMENT=lost_indices)
>

>
>
>
>
> Depending on what your purpose is, you might not even need to loop over j. You could just do
>
>
>
> keep_indices = where((zall ne -32768.0) and (zall gt 10), COMPLEMENT=lost_indices)
>
> zall[lost_indices] = !values.f_nan

Thanks for your help, I think what you are saying would have worked. In the meantime, however, I realized my methodology was inefficient for my purpose and rewrote it completely. I was able to reach a satisfactory end result.
