Subject: Re: Collection of different size arrays?
Posted by David Fanning on Tue, 13 Jul 2004 22:54:49 GMT
View Forum Message <> Reply to Message

Jonathan Greenberg writes:

> I'm unfamiliar with the use of structures, but I'm guessing this is the
> answer.

No, when you don't understand something, the answer is always "objects". :-)

In your case, it may be "pointers".

```
> Lets say I have some program generate arrays using the following
> algorithm:
>
> For I=1,5 do begin
>    temparr=fltarr(i)
> Endfor
>
> I want to "collect" all of the temparr that are created within the loop such
> that I can access each array by some sort of index (which would be from 0 to
> 4) -- e.g. I want to be able to do something like:
>
> A = temparr at index 2
> Print,A
> --> (0,1)
>
> An obvious solution is to put "0s" in and make a 2d array, where the row is
> the index, so I get something like
>
> 0,0,0,0,0
> 0,1,0,0,0
> 0,1,2,0,0
> 0,1,2,3,0
> 0,1,2,3,4
>
> However, this takes up a lot of space and the nature of the algorithm I want
> to use I would like to have smaller arrays be "compressed" without having to
> crop them each time.
>
> What do I put inside the for loop to generate:
> 1:0
> 2:0,1
> 3:0,1,2
> 4:0,1,2,3
> 5:0,1,2,3,4
```

>
> ???

How about this:

```
IDL>   a = PtrArr(5)
IDL>   FOR j=0,4 DO a[j] = Ptr_New(Indgen(j+1), /No_Copy)
IDL>   FOR j=0,4 DO Print, StrTrim(j+1) + ': ', *a[j]
     1:       0
     2:       0       1
     3:       0       1       2
     4:       0       1       2       3
     5:       0       1       2       3       4
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Collection of different size arrays?
Posted by cedricl on Wed, 14 Jul 2004 02:11:21 GMT
View Forum Message <> Reply to Message

Jonathan Greenberg <greenberg@ucdavis.edu> wrote in message
news:<BD19AC1F.1052D%greenberg@ucdavis.edu>...
> I'm unfamiliar with the use of structures, but I'm guessing this is the
> answer. Lets say I have some program generate arrays using the following
> algorithm:

Pointers are probably the cleanest solution. But personally I don't
like having to care about possible memory leaks, so here are two
alternatives:

1. Use a sparse matrix. Look up the SPRS* functions in the help. I'm
not sure how easy it is to access individual elements, since they were
designed for matrix operations, and not really storage. Perhaps you
could build a simple function that takes care of the indexing.

2. Use structures. This is an ugly and inefficient solution, but it
can be done, and the uglyness could be contained in a few key
functions. The fields of a structure can be accessed by an index:
"print, some_structure.(5)", and of course, each field of a structure
can be an array of a different size. These fields can be built

automatically using create_structure. I can flesh out the details
tomorrow if you want.

Anyway, unless you have some profound aversion to pointers, like I do,
you should use them in this case.

BTW, is there an online version of the IDL docs?

Cedric