Subject: Self-sizing arrays (was array concat and opt)
Posted by ngls on Thu, 04 Oct 2001 14:22:42 GMT
View Forum Message <> Reply to Message

shaugan@esa.nascom.nasa.gov (Stein Vidar Hagfors Haugan) wrote in
<xmzsnd7ju50.fsf@esa.nascom.nasa.gov>:

> If you're taking the trouble of hiding it all inside an object, why
> not go further to use e.g. lists, dropping the need for replacing
> anything until possibly after as part of a "reconstitution" operation
> (would need the user program to signal when building is finished - or
> possibly trigger it automatically when a first read access is made ?)
>

I've also written a "self sizing array" (Vector) class very similar to
Mark's.

(For those not in the know, C++ and Java - at least - both offer self
sizing array classes (objects) called Vectors - hence the use of the term.
My class is similarly called c_vector. This does not mean the arrays must
be 1-dimensional, as the name might suggest!)

To answer Stein's question, I did consider writing it using lists, but to
me it was important to be able to access array slices of the data stored in
the 'vector'. For this reason I store all the data in a single array (with
an arbitrary number of dimensions). Whilst access to single array elements
causes no change to the internal array size, if the user requests all the
data (either a pointer or copy of the whole array) it is trimmed to size.
This allows the user to get slices of the array. It also works with arrays
of structures.

To answer the orginal question (but not with structures) you could do
something like this:

```
PRO test_vector

 file = '5_cols_test.txt'
 size_guess = 500
 capacity_incr = 101

 line = FLTARR(5) ;Each line in file contains 5 floats
 data = OBJ_NEW('c_vector', line, size_guess, capacity_incr)

 OPENR, lun, file, /GET_LUN

 WHILE NOT(EOF(lun)) DO BEGIN
  READF, lun, line
  data -> ADD_ELEMENT, line
```

```
ENDWHILE

FREE_LUN, lun ;Free lun and close

PRINT, 'Numbers of lines read in:', data -> GET_SIZE() ;Displays 1000
r = data -> GET_DATA_REF()
PRINT, 'Dimensions of data array:', SIZE(*r, /DIM) ;Displays 5 1000
PRINT, 'Mean of third column:', MEAN( (*r)[2,*] ) ;Displays 0.504

OBJ_DESTROY, data

END
```

If anyone is interested I can post the c_vector code...

Justin
