View Forum Message <> Reply to Message Am Montag, 29. Juli 2013 11:44:14 UTC+2 schrieb fawltyl...@gmail.com: > On Monday, July 29, 2013 10:46:36 AM UTC+2, mschellens wrote: > > >> As of IDL 8.0, this is not correct. An IDL LIST is really a single linked list made up of (PTR) heap variable nodes (IDL CONTAINER NODE). The IDL CONTAINER::GET function creates then the array. > >> >> But your method works, as the (copied) pointers access the same heap variables. This is also the core of the mechanism I suggested for OVERLOADBRACKETSLEFTSIDE. > >> >> Also note, that at least with HEAP the IDL CONTAINER::GET functionality cannot work anymore (as you cannot pick the right element). > >> >> And it is of course as well not efficient, to convert the complete container to a pointer array in order to left-access one element. >> >> And the call to GET is almost as ugly as copying out one element, left-accessing it and copying it back. > >> > > > With huge list elements, copying out and back is very unefficient, creating a pointer array is much faster. > > > I do not understand the idea behind list. If it is a linked list, then accessing elements through subscripting is O(n) vs. O(1) in arrays. This makes lists practically unusable. > > Try this test program to access the last element in a list:

Subject: Re: Modifying Arrays and Structures in HASH's (hint: you can't)

Posted by m\_schellens on Mon, 29 Jul 2013 11:24:01 GMT

```
>
>
> pro list_test
> I=list(1,2,3)
>
>
> t=systime(1)
> for j=1,10l^6 do x=l[-1]
> print, " 3 elements: ", systime(1)-t
>
>
> for j=4,10l<sup>3</sup> do l.add, j
>
> t=systime(1)
> for j=1,10l^6 do x=l[-1]
> print, " 10^6 elements: ", systime(1)-t
>
>
>
> end
>
> IDL 8.2.3:
>
>
  3 elements:
                    7.6518829
                        47.781787
   10^6 elements:
>
>
> GDL CVS:
>
>
  3 elements:
                    0.5020251
```

```
>
>
   10^6 elements:
                       44.200854
>
>
>
> FL 0.79.25:
>
>
  3 elements:
                  0.044948101
>
   10^6 elements:
                      0.042613983
>
>
>
>
> My pointer array based LIST implementation is about 10-150x faster for the small list, and more
than 1000x faster for the large list.
>
>
>
> regards,
>
> Lajos
```

I would like to emphasize, that I revided this thread for the suggestion about \_OVERLOADBRACKETSLEFTSIDE. This is not limited to a particular container type. What do you think about it?

The strength of a LIST is the deletion and insertion of elements.

Particular at the beginning or at the end (O(1)).

Not the traversal, what you measured.

I am sure, one can build an example, where a list implementation based on an array will loose against a real linked list. What if you fill the complete LIST from the left (like:

list.ADD,element[i],0)?

For an array based LIST, even as you demonstrate that it is for some cases more efficient, one could say: Why not using a PTR array then directly?

Ok, you got some comfort functions. Maybe there is even room (or need) for an array based container with ADD, REMOVE, ....

But I think if the user uses a LIST he possibly really want one.

Regards, Marc