
Subject: Re: LIST extensions

Posted by [Matt Haffner](#) on Wed, 15 Dec 2010 06:51:32 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Dec 14, 3:51 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:

> Hello,

>

> Sorry to keep flogging this particular topic, but I'm currently up to my eyes in adapting some code from using pointers

> to using lists (partly for curiosity, partly to avoid nested pointer dereferencing), and I've found the very simple

> methods below to be useful in dealing with the "is it empty?", "are there nulls in the list?" etc type of questions I've

> been asking here.

>

> Comments?

>

> cheers,

>

> paulv

>

> ;+

> ; NAME:

> ; List::Empty

> ;

> ; PURPOSE:

> ; The List::Empty function method return TRUE if a list contains
> ; no elements, FALSE otherwise.

> ;

> ; CALLING SEQUENCE:

> ; result = Obj->[List::]Empty()

> ;-

> FUNCTION List::Empty

> RETURN, N_ELEMENTS(self) EQ 0

> END

>

> ;+

> ; NAME:

> ; List::Length

> ;

> ; PURPOSE:

> ; The List::Length function method returns the number of
> ; elements in a list.

> ;

> ; CALLING SEQUENCE:

> ; result = Obj->[List::]Length()

> ;-

> FUNCTION List::Length

```

> RETURN, N_ELEMENTS(self)
> END
>
> ;+
> ; NAME:
> ; List::n_Items
> ;
> ; PURPOSE:
> ; The List::n_Items function method returns the number of
> ; non-null elements in a list.
> ;
> ; CALLING SEQUENCE:
> ; result = Obj->[List::]n_Items()
> ;-
> FUNCTION List::n_Items
> idx = WHERE(self NE !NULL, count)
> RETURN, count
> END
>
> ;+
> ; NAME:
> ; List::Compact
> ;
> ; PURPOSE:
> ; The List::Compact function method returns a copy of a list
> ; with all null elements removed.
> ;
> ; CALLING SEQUENCE:
> ; result = Obj->[List::]Compact()
> ;-
> FUNCTION List::Compact
> idx = WHERE(self NE !NULL, count)
> RETURN, ( count GT 0 ) ? self[idx] : LIST()
> END

```

Useful--thanks.

Also, note that LIST is a subclass of IDL_Container, which has a .Count() method, so .Length() may not be needed. There is also a .Move method to rearrange items in a container. Unfortunately the .IsContained method doesn't seem to work for me on a LIST though (and is only for objects, in any case):

```

IDL> z=g[50]
IDL> print,g.IsContained(z)
      0
IDL> print,obj_valid(z, /get_heap)
1884104

```

```
IDL> print,obj_valid(g[50], /get_heap)
```

```
1884104
```

```
IDL> print, z eq g[50]
```

```
1
```

```
mh
```
