Subject: Re: Top 10 IDL Requests
Posted by John-David T. Smith on Thu, 10 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Otte Homan (ottehoman@my-deja.com) writes:
>
>> I have the same problem - I'd like to create a
>> hierarchical data structure, basically an array,
>> of structures (so I can use DataFile(1),
>> DataFile(2), etc...) Each structure consists of
>> headers, parameters, and *dynamic* arrays. So the
>> structures are *more or less* the same, but
>> differ in the lenght of their arrays. Using IDL
>> 5.3 (sorry, our site has only this
>> version licensed) the contents of an array can
>> only be of one single type. My filestructures
>> are like this: {{header},{parameters},{data}},
>> where {data} is a structure with arrays of (from
>> file to file) different lengths. I only know at
>> runtime how long these arrays are.
>>
>> Any solution ?
>
> Well, I reiterate. Pointers. The solution is pointers. :-)
>
> If the data field of this structure is a pointer to
> the variable length array, then you can store as many
> of these structures in an array as you like.
>
> William asked for an array of different structures,
> which sort of turns the definition of an array
> topsy-turvy. But *this* problem can be solved with
> pointers, I'm sure of it. :-)
>
> Cheers,
>
> David

I have sensed some great hesitation over the use of pointers for complex date structures. To ease the feeling that you'll be lost in a maze of no return, I post here a summary of a single data structure of mine, which, while at first glance unwieldy, is actually quite flexible and reasonably easy to use. To summarize:

INHERIT'ing object class scoreProj containing:

1. various "regular" numerical and string data member fields.
2. pointer to a dynamic list of struct of type SCORE_DR
 SCORE_DR containing:
 a. Various regular fields.
 b. Pointer to a dynamically sized array of strings (filenames)
 c. Pointer to a data array of size 128x128xn (n determined at runtime)
 d. Pointer to a data array of size 128x128x2
 e. Pointer to dynamically sized list of planes.
 f. Pointer to dynamic array of pointers to dynamic string arrays.
3. pointer to a struct of type SCORE_STACK
 SCORE_STACK containing:
 a. Various regular fields
 b. 4 pointers to data arrays of size 128x128x2
 d. Pointer to dynamic list of floating pairs (2xn)
4. pointer to a struct of type SCORE_EXTRACT
 SCORE_EXTRACT containing:
 a. Various regular fields
 b. 3 pointers to data of size 128x128
 c. A pointer to data of size 3xn
5. pointer to a struct of type scoreProj_wInfo
 scoreProj_wInfo containing:
 a. Various widget id's as longs.
 b. Pointer to dynamic array of button id's.

This doesn't even go into the inherited data members.

The deepest reference is in 2f. and goes something like

```
*(*(*self.DR)[i].HEADER)[j]
```

which may frighten you, but I assure you after a bit of practice and review of
the (unwritten) precedence of "*", is quite tractable.

Happy pointering,

JD

```
--
 J.D. Smith                          /*\    WORK: (607) 255-6263
 Cornell University Dept. of Astronomy  \*/      (607) 255-5842
 304 Space Sciences Bldg.               /*\     FAX: (607) 255-5875
 Ithaca, NY 14853                     \*/
```