Subject: Re: Accessing 2D array from pointer array within structure
Posted by David Fanning on Thu, 15 Aug 2002 16:25:29 GMT
View Forum Message <> Reply to Message

Brandon Jones (bjones8@yahoo.com) writes:

> I have a structure called "info" with an item called "img" which is a
> pointer array. Each of these pointers point to an image (2D array).
>
> IMG POINTER Array[5]
>
> I want to be able to access the image via subscripts. Is it possible?
> What is the syntax?
>
> So something like
> value=*info.img[3][130,453]
>
> I know this is invalid....but something along those lines. Right now
> my code does this, which is sloppy:
>
> tmp=*info.img[3]
> value=tmp[130,453]
>
> I can't really find any info about doing this in the docs....

The correct syntax is:

```
value=(*(info.img[3]))[130,453]
```

The problem you have here is that pointer de-referencing
has the very lowest order of precedence, lower than
array subscripting. The parentheses have the highest order
of precedence, so we throw them around pretty liberally
when we want to get things like this to work. :-)

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Accessing 2D array from pointer array within structure

Posted by bjones8 on Thu, 15 Aug 2002 22:46:19 GMT
View Forum Message <> Reply to Message

David--

Thanks again! It didnt occur to be that it was a operator precedence issue...now my program is alot faster :-)

-Brandon

David Fanning <david@dfanning.com> wrote in message news:<MPG.17c583df8c8d86a5989959@news.frii.com>...
> Brandon Jones (bjones8@yahoo.com) writes:
>
>> I have a structure called "info" with an item called "img" which is a
>> pointer array. Each of these pointers point to an image (2D array).
>>
>> IMG   POINTER   Array[5]
>>
>> I want to be able to access the image via subscripts. Is it possible?
>> What is the syntax?
>>
>> So something like
>> value=*info.img[3][130,453]
>>
>> I know this is invalid....but something along those lines. Right now
>> my code does this, which is sloppy:
>>
>> tmp=*info.img[3]
>> value=tmp[130,453]
>>
>> I can't really find any info about doing this in the docs....
>
> The correct syntax is:
>
>   value=(*(info.img[3]))[130,453]
>
> The problem you have here is that pointer de-referencing
> has the very lowest order of precedence, lower than
> array subscripting. The parentheses have the highest order
> of precedence, so we throw them around pretty liberally
> when we want to get things like this to work. :-)
>
> Cheers,
>
> David

Subject: Re: Accessing 2D array from pointer array within structure
Posted by JD Smith on Fri, 16 Aug 2002 20:24:11 GMT
View Forum Message <> Reply to Message

On Thu, 15 Aug 2002 09:25:29 -0700, David Fanning wrote:

> Brandon Jones (bjones8@yahoo.com) writes:
>
>> I have a structure called "info" with an item called "img" which is a
>> pointer array. Each of these pointers point to an image (2D array).
>>
>> IMG POINTER Array[5]
>>
>> I want to be able to access the image via subscripts. Is it possible?
>> What is the syntax?
>>
>> So something like
>> value=*info.img[3][130,453]
>>
>> I know this is invalid....but something along those lines. Right now
>> my code does this, which is sloppy:
>>
>> tmp=*info.img[3]
>> value=tmp[130,453]
>>
>> I can't really find any info about doing this in the docs....
>
> The correct syntax is:
>
> value=(*(info.img[3]))[130,453]
>
> The problem you have here is that pointer de-referencing has the very
> lowest order of precedence, lower than array subscripting. The
> parentheses have the highest order of precedence, so we throw them
> around pretty liberally when we want to get things like this to work.
> :-)
>
> Cheers,
>
> David

That's really more parens than you need:

value=(*info.img[3])[130,453]

I don't blame you though. One of my major complaints about the IDL
documentation is that the operator precendence table given leaves out
both structure dereference "." and array indexing "[]" operators.
I've alerted them to the problem, and hopefully they'll get around to

fixing it. As you can see, "[]" is of higher precedence than "*", and "." has the highest precedence of all (except for "()"). What if info had been a list of structures with an image pointer?

```
IDL> info1={img:ptrarr(5)}
IDL> print,size(info1.img,/DIMENSIONS)
          5
IDL> info2=replicate({img:ptr_new()},5)
IDL> print,size(info2.img,/DIMENSIONS)
          5
```

OK, these are exactly the same size and type, but puzzle over this:

```
IDL> print,info1.img[0]
<NullPointer>
IDL> print,info2.img[0]
<NullPointer><NullPointer><NullPointer><NullPointer><NullPointer >
IDL> print,info2.img[1]
% Subscript range values of the form low:high must be >= 0, < size, with
low <= high: <No name>.
```

What's going on here? Well, info2 is an array, and info1 is a scalar. This appears to matter. The correct notation is:

```
IDL> print,info2[1].img
<NullPointer>
```

We index *before* we use the structure dereference. This is a funny kind of precedence, but you need to keep it in mind. This is more efficient than the alternative:

```
IDL> print,(info2.img)[1]
<NullPointer>
```

and you can dereference simply too:

```
IDL> info2[1].img=ptr_new(dist(10))
IDL> print,*info2[1].img
      0.00000      1.00000      2.00000    ....
```

My rules of thumb:

1. Always index as close as possible to the array being indexed (e.g. "info2[1].img").
2. "()" -> "." -> "[]" -> "*" is the precedence order, subject to Rule #1 (which can swap the order of the inner two).

Pointer de-reference has relatively weak precedence, and often needs the help of parentheses to achieve what you're after. Thus:

(*a.b)[42] for the 43rd element of the list pointed to by the pointer in field "b" of structure "a".

*a[42].b for the value pointed to by the pointer in field "b" of the 43rd structure in the list of structures "a".

(*a.b)[42].c for the value of field "c" of the 43rd structure in the list of structures pointed to by the pointer in field "b" of structure "a".

Etc. Here's one to puzzle out:

(*(*a.b[4])[40].c)[2]

At some level of complexity, it becomes easier to break this up into multiple calls, or redesign your data format to be simpler to begin with.

JD
