Subject: Re: Structure Containing Structure: question about parentheses
Posted by David Fanning on Thu, 09 Aug 2007 17:20:33 GMT
View Forum Message <> Reply to Message

On Aug 9, 9:36 am, lark...@gmail.com wrote:

```
>  I have a question about using parentheses with structures that are
>  members of a structure.  I include a sample program below that
>  demonstrates the behaviors.
>
>  Basically I have no problem if I run the command
>
>  state.str.c = 50
>
>  But if I run the command
>
>  (state.str).c = 50
>
>  then I  get the error:
>
>  Attempt to store into an expression: Structure reference.
>  Execution halted at: TESTSTRUCT        32 C:\teststruct.pro
>                       $MAIN$
>
>  Can someone explain this behavior to me?
```

Here is an article that explains what is going on:

  http://www.dfanning.com/misc_tips/ptr_struct.html

Cheers,

David

Subject: Re: Structure Containing Structure: question about parentheses
Posted by JD Smith on Thu, 09 Aug 2007 18:45:53 GMT
View Forum Message <> Reply to Message

On Thu, 09 Aug 2007 08:36:34 -0700, larkn10 wrote:

```
>  Hi All,
>
>  I have a question about using parentheses with structures that are
>  members of a structure.  I include a sample program below that
>  demonstrates the behaviors.
>
>  Basically I have no problem if I run the command
```

>
> state.str.c = 50
>
> But if I run the command
>
> (state.str).c = 50
>
> then I get the error:
>
> Attempt to store into an expression: Structure reference.
> Execution halted at: TESTSTRUCT 32 C:\teststruct.pro
> $MAIN$
>
>
> Can someone explain this behavior to me?

This is the exact analog of being unable to pass a structure member by reference. (state.str) creates a temporary variable, a copy of the str structure inside of state. If you assign to that, IDL is smart enough to realize you don't want to change the temporary copy, but the real one, as if you hadn't written the parens. Once you add a further structure dereference, however, IDL stops saving you from yourself. Since you can't assign to temporary variables, you get the error.

Not only is the above paren set superfluous, if it worked, it would be much slower, since it creates an unnecessary copy of the structure directly. Since it does work when used for anything but assignment, you should avoid it.

Parentheses are useful and needed when you have nested pointers inside of structures; see the operator precedence tutorial.

JD

Subject: Re: Structure Containing Structure: question about parentheses
Posted by larkn10 on Thu, 09 Aug 2007 21:12:07 GMT
View Forum Message <> Reply to Message

On Aug 9, 2:45 pm, JD Smith <jdsm...@as.arizona.edu> wrote:
> On Thu, 09 Aug 2007 08:36:34 -0700, larkn10 wrote:
>> Hi All,
>
>> I have a question about using parentheses with structures that are
>> members of a structure. I include a sample program below that
>> demonstrates the behaviors.
>
>> Basically I have no problem if I run the command

```
>
>>  state.str.c = 50
>
>>  But if I run the command
>
>>  (state.str).c = 50
>
>>  then I  get the error:
>
>>  Attempt to store into an expression: Structure reference.
>>  Execution halted at: TESTSTRUCT        32 C:\teststruct.pro
>>                      $MAIN$
>
>>  Can someone explain this behavior to me?
>
>  This is the exact analog of being unable to pass a structure member by
>  reference.  (state.str) creates a temporary variable, a copy of the
>  str structure inside of state.  If you assign to that, IDL is smart
>  enough to realize you don't want to change the temporary copy, but the
>  real one, as if you hadn't written the parens.  Once you add a further
>  structure dereference, however, IDL stops saving you from yourself.
>  Since you can't assign to temporary variables, you get the error.
>
>  Not only is the above paren set superfluous, if it worked, it would be
>  much slower, since it creates an unnecessary copy of the structure
>  directly.  Since it does work when used for anything but assignment,
>  you should avoid it.
>
>  Parentheses are useful and needed when you have nested pointers inside
>  of structures; see the operator precedence tutorial.
>
>  JD
```

Thanks David and JD! That explains the unexpected behavior!