Subject: Re: Functions and arrays
Posted by thompson on Thu, 05 Dec 1996 08:00:00 GMT
View Forum Message <> Reply to Message

Peter Mason <peterm@demsyd.syd.dem.csiro.au> writes:

> On 4 Dec 1996, Stein Vidar Hagfors Haugan wrote:
>> People at RSI (and not just support people) do read this newsgroup (and
>> act on it as well: XPAD/YPAD/SPACE=0 will once again rule, in IDL v 5.0!),
>> so this place may very well be the best place for a campaign to remove
>> this quite serious (and dangerous) "feature". I do think that we need to
>> show some enthusiasm to make it happen, though (i.e., the more people
>> agreeing about this being a rather nasty thing that they'd like to see
>> removed, the better).

> I'd also really like to see this ambiguity sorted out.

>> Or at least get a compile time error about the possible mixup,
>> something like "Error: test1 interpreted as a function in line 5,
>> but as a variable in line 10".

> I think that this would be a good idea. It wouldn't bust any code that
> wasn't already hovering on the edges of "busthood", and it would catch many
> of the ambiguities. When requested to compile a function, IDL could stop with
> an error if a variable of the same name already existed. ...

There seems to be a assumption here that the *PROGRAMMER* is forming an ambiguity by trying to use the same name for both a variable and a function in the same routine. I argue that it's *IDL* which is responsible for the ambiguity. The situation I ran into was when the software was written in a self-consistent manner--the name was intended to refer to a variable throughout the routine. However, IDL on its own decided to sometimes interpret the call as a variable (correct) and sometimes as a function (incorrect), depending on what applications were started first in the IDL session.

The correction to this is quite simple. If a name is used for a variable in a subroutine, then it refers to that variable throughout that subroutine. The fact that it may be used to refer to a function in some other routine is completely immaterial. I'm sure that the IDL syntax parser could be made to be unambiguous here--as somebody pointed out it just requires a double pass through the source code.

The idea that a correctly written and working piece of code could be broken by adding a function in a completely unrelated piece of code is totally repugnant to me.

Bill Thompson