Subject: Re: Declaration of variables in IDL
Posted by David Fanning on Wed, 03 Oct 2001 13:21:08 GMT
View Forum Message <> Reply to Message

Hans de Boer (boer@kfih.azr.nl) writes:

> I have been working with IDL for quite a while now, but never
> encountered a way of fixing variable types in IDL. Especially when
> adding new variables to old pieces of code, it can be very confusing if
> a variable of that name was already in use. IDL simply changes the
> variable type (including changes of array dimension) if required and
> possible. If not possible, you are the lucky one to get run-time errors,
> but if possible, the code seems to run quite nicely and the poor
> programmer is left with apparently correct results.
> I also have found that people new to IDL have generated extremely sneaky
> bugs this way.
> Does anybody know of making IDL performing an explicit variable check
> during compilation ?

I think you need to switch to Pascal. :-)

IDL, as a weakly-typed language, is inherently
messy. That's the downside. The upside is that
it is also powerful for this very reason. No
question you can get into trouble if you are
not careful, but this applies to love as well
as software, and we find a way to make do.

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
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: Declaration of variables in IDL
Posted by Paul van Delst on Wed, 03 Oct 2001 14:28:33 GMT
View Forum Message <> Reply to Message

Hans de Boer wrote:
>
> Hi there,
>

> I have been working with IDL for quite a while now, but never
> encountered a way of fixing variable types in IDL. Especially when
> adding new variables to old pieces of code, it can be very confusing if
> a variable of that name was already in use. IDL simply changes the
> variable type (including changes of array dimension) if required and
> possible. If not possible, you are the lucky one to get run-time errors,
> but if possible, the code seems to run quite nicely and the poor
> programmer is left with apparently correct results.
> I also have found that people new to IDL have generated extremely sneaky
> bugs this way.
> Does anybody know of making IDL performing an explicit variable check
> during compilation ?

I dunno about IDL, but my immediate solution (after checking in all the code to be altered into a version control system) would be to sit the prospective programmers in front of the code with a pad and a pencil and disable all but the editor-arrow keys and editor-search functions (to poll the namespace for variables names) and give them a day or two to think about what they need to do, how they would go about doing it (more than one method would be a good thing), and what are the consequences of each method. This initial sit down and think about it period will save a lot more time later because the code won't fall in the messy heap that usually results with the tinker-and-recompile method of change/debugging.

I have had people change some of my code and then later (after they forgot what they did) complain that my code doesn't work and is broken (and sometimes, whimper...sniff, they're not that polite). Needless to say a couple of wasted hours later the code works again - usually by me doing a "rm -fr *" (== delete absolutely everything for you windows/mac fellers) followed by a "cvs checkout".

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Declaration of variables in IDL
Posted by Todd Clements on Wed, 03 Oct 2001 16:03:44 GMT
View Forum Message <> Reply to Message

> Does anybody know of making IDL performing an explicit variable check
> during compilation ?

I don't think there is a way to do this since, as David said, IDL is a weakly typed language.

However, you can get *some* type of variable checking during run time by

using structures.

a = {b: fltarr(10), c: 0L}

a.b = 5 ;; this works, setting all elements of b to 5
a.b = 'hi' ;; this DOESN'T work, yielding a type conversion error
a.b = findgen(7) ;; this does work, yielding 0,1,2,3,4,5,6,0,0,0
a.c = [2,3] ;; this DOESN'T work since C is a scalar

Even though you can get some rudimentary type checking this way, I think it can cause more problems than it is worth since cases like the third one might give unexpected results that you don't detect later. It's much better to embrace the weakly typed variables and reuse the same variable several times in each function. That way, your job prospects are much more secure since you're the only one who can figure out how your code works (my background and training in C programming is coming out now, I think...)

Todd

Subject: Re: Declaration of variables in IDL
Posted by David Fanning on Wed, 03 Oct 2001 16:08:56 GMT
View Forum Message <> Reply to Message

Todd Clements (mole6e23@hotmail.com) writes:

> However, you can get *some* type of variable checking during run time by
> using structures.
>
> a = {b: fltarr(10), c: 0L}
>
> a.b = 5 ;; this works, setting all elements of b to 5
> a.b = 'hi' ;; this DOESN'T work, yielding a type conversion error
> a.b = findgen(7) ;; this does work, yielding 0,1,2,3,4,5,6,0,0,0
> a.c = [2,3] ;; this DOESN'T work since C is a scalar
>
> Even though you can get some rudimentary type checking this way, I think
> it can cause more problems than it is worth...

With the exception of strings, I don't think I would call this "type" checking at all. More like "space" checking, since IDL doesn't care what you stuff into a structure field as long as it fits into the memory set aside for it.

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting
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: Declaration of variables in IDL
Posted by Paul van Delst on Wed, 03 Oct 2001 16:09:40 GMT
View Forum Message <> Reply to Message

Todd Clements wrote:
>
> It's much
> better to embrace the weakly typed variables and reuse the same variable
> several times in each function.

Note to original poster: Apologies for maybe stating the obvious, but Todd Clements is joking. If you can avoid it, you should not reuse the same variable several times in each function - it can be a maintenance nightmare depending on the complexity of the function/procedure in question.

paulv

p.s. If he's not joking, he should be :o)

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Declaration of variables in IDL
Posted by R.G.S. on Wed, 03 Oct 2001 16:10:22 GMT
View Forum Message <> Reply to Message

Hans de Boer <boer@kfih.azr.nl> wrote in message
news:3BBB0EC8.1CCB5E11@kfih.azr.nl...
> Hi there,
>
> I have been working with IDL for quite a while now, but never
> encountered a way of fixing variable types in IDL. Especially when
> adding new variables to old pieces of code, it can be very confusing if

> a variable of that name was already in use. IDL simply changes the
> variable type (including changes of array dimension) if required and
> possible. If not possible, you are the lucky one to get run-time errors,
> but if possible, the code seems to run quite nicely and the poor
> programmer is left with apparently correct results.
> I also have found that people new to IDL have generated extremely sneaky
> bugs this way.
> Does anybody know of making IDL performing an explicit variable check
> during compilation ?
>
> Thanks
>
> Hans deB

As answered, IDL allows type changing. Such is life.

This may be too simple to help you, but your function can
explicitly check variable type. For instance:

function foo_wrapper,dnumber,fnumber,stringvar

if size(dnumber,/type) ne 5 then message,'ERROR: DNUMBER type changed or
undefined.'
if size(fnumber,/type) ne 4 then message,'ERROR: FNUMBER type changed or
undefined.'
if size(stringvar ,/type) ne 7 then message,'ERROR: STRINGVAR type changed
or undefined.'
return,realfunction_foo(dnumber,fnumber,stringvar)
end

Cheers,
bob

Subject: Re: Declaration of variables in IDL
Posted by Pavel A. Romashkin on Wed, 03 Oct 2001 16:14:08 GMT
View Forum Message <> Reply to Message

Hans de Boer wrote:
>
> the code seems to run quite nicely and the poor
> programmer is left with apparently correct results.
> I also have found that people new to IDL have generated extremely sneaky
> bugs this way.

This is a good example of what you can use the sophisticated system of
breakpoints that IDL comes with. Debugging is convenient and simple. The

only thing not there is an automatic coffe pot that would brew any time
a bug is found :-)
As a universal method, you could check SIZE, /TYPE every time you use a
variable. Never fails, but may double your code length when used on
every second line :-)

Cheers,
Pavel

Subject: Re: Declaration of variables in IDL
Posted by David Fanning on Wed, 03 Oct 2001 16:22:29 GMT
View Forum Message <> Reply to Message

Paul van Delst (paul.vandelst@noaa.gov) writes:

> Note to original poster: Apologies for maybe stating the obvious, but Todd Clements is joking.
> If you can avoid it, you should not reuse the same variable several times in each function - it
> can be a maintenance nightmare depending on the complexity of the function/procedure in
> question.

Todd was talking about job security, so I am sure
he was not joking. With recent events and today's
job economy, it is no joking matter. I have renamed
almost all the variables in my programs "a" to cope.

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting
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: Declaration of variables in IDL
Posted by Paul van Delst on Wed, 03 Oct 2001 16:30:45 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
>
> Paul van Delst (paul.vandelst@noaa.gov) writes:
>
>> Note to original poster: Apologies for maybe stating the obvious, but Todd Clements is joking.
>> If you can avoid it, you should not reuse the same variable several times in each function - it

>> can be a maintenance nightmare depending on the complexity of the function/procedure in
>> question.
>
> Todd was talking about job security, so I am sure
> he was not joking. With recent events and today's
> job economy, it is no joking matter. I have renamed
> almost all the variables in my programs "a" to cope.

Sigh - I guess I'd better restore my IDL/Fortran program variable-name randomiser script from backup tape then. It also has optional arguments to strip all comments and "de-format" any prettifying structure to the code that may (*gasp*) make it easy to read/understand.

I'll sic that puppy onto my distributed code, encrypt, uuencode, print, then delete the resulting variable match lookup tables. I think there's enough room in my safe deposit box for all that printout.....

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Declaration of variables in IDL
Posted by Paul van Delst on Wed, 03 Oct 2001 16:32:09 GMT
View Forum Message <> Reply to Message

"Pavel A. Romashkin" wrote:
>
> Hans de Boer wrote:
>>
>> the code seems to run quite nicely and the poor
>> programmer is left with apparently correct results.
>> I also have found that people new to IDL have generated extremely sneaky
>> bugs this way.
>
> This is a good example of what you can use the sophisticated system of
> breakpoints that IDL comes with. Debugging is convenient and simple. The
> only thing not there is an automatic coffe pot that would brew any time
> a bug is found :-)
> As a universal method, you could check SIZE, /TYPE every time you use a
> variable. Never fails, but may double your code length when used on
> every second line :-)

It tripled mine until I discovered the wonders of CATCH. :o)

paulv

--
Paul van Delst Religious and cultural
CIMSS @ NOAA/NCEP purity is a fundamentalist
Ph: (301)763-8000 x7274 fantasy
Fax:(301)763-8545 V.S.Naipaul

Subject: Re: Declaration of variables in IDL
Posted by Todd Clements on Wed, 03 Oct 2001 19:49:39 GMT
View Forum Message <> Reply to Message

I get ahead of myself sometimes on the sarcasm. In my family, we first
assume that what you say is a joke. Then we consider whether or not it
may be true. Sometimes this get me in trouble. ;>

But like I said, using the same variable name many times is great for
job security. Or at least guaranteeing that you'll be bugged for years
to come. Someone wrote a monolithic code a while back which he called
"SDC" standing for "self documenting code". This was because some of his
variables actually had *gasp* more than seven characters. Let's just say
(TM) that for a while after I started working on the code, he was on
speed dial.

Todd

>> It's much
>> better to embrace the weakly typed variables and reuse the same variable
>> several times in each function.
>
> Note to original poster: Apologies for maybe stating the obvious, but Todd
> Clements is joking.
> If you can avoid it, you should not reuse the same variable several times in
> each function - it
> can be a maintenance nightmare depending on the complexity of the
> function/procedure in
> question.

Subject: Re: Declaration of variables in IDL
Posted by Craig Markwardt on Wed, 03 Oct 2001 21:43:46 GMT
View Forum Message <> Reply to Message

"R.G.S." <rgs1967@hotmail.com> writes:

> Hans de Boer <boer@kfih.azr.nl> wrote in message

> news:3BBB0EC8.1CCB5E11@kfih.azr.nl...
>> Hi there,
>>
>> I have been working with IDL for quite a while now, but never
>> encountered a way of fixing variable types in IDL. Especially when
>> adding new variables to old pieces of code, it can be very confusing if
>> a variable of that name was already in use. IDL simply changes the
>> variable type (including changes of array dimension) if required and
>> possible. If not possible, you are the lucky one to get run-time errors,
>> but if possible, the code seems to run quite nicely and the poor
>> programmer is left with apparently correct results.
>> I also have found that people new to IDL have generated extremely sneaky
>> bugs this way.
>> Does anybody know of making IDL performing an explicit variable check
>> during compilation ?
>>
>> Thanks
>>
>> Hans deB
>
> As answered, IDL allows type changing. Such is life.

I agree 100%. The advantage of IDL is that the type of the variable can change dynamically. The disadvantage is that the type of the variable can change dynamically.

My approach is two-fold. First, test any input variables to a procedure to ensure they are the correct type AND DIMENSION to the ones you were expecting. If these attributes aren't correct then there are two options. Either error out, or try to recast the data to the correct type and dimension.

So, for example, if I am expecting a parameter to be a long integer scalar, then I usually cast it myself doing this:

```
param = round( param(0) )
```

This does two things, it ensures that param is a scalar, by scooping off the first element, and it also ensures that PARAM is cast to a long integer. The same thing applies when casting to strings, as in param = strtrim(param(0), 2)

Then things can get even more sticky, because you also need to decide whether you are going to modify the caller's original argument, or make a copy for yourself. Modifying the caller's argument can confuse your caller sometimes. It's usually not polite. For example, I usually modify the above approach to make a copy. The original parameter is PARAM0, and the internal copy is called PARAM.

```
  param = round( param0(0) )
```

Also it is crucial to document the expected type and dimension of each parameter. That way the user can know ahead of time what to pass. If they pass the wrong thing then the blame can be assigned to them rather than you :-)

***

The second thing is to follow type-safe practices within your procedures. I am sure other people can come up with some good ideas. Here are some of mine.

The output of TOTAL is always a floating point type. Beware of this if you operate on integers and expect TOTAL to output an integer!

Beware of constructing an output variable by doing OUT = FLTARR(N). The problem with this is, what if the user is using double precision? You have just ruined their precision! Better to do something like this: OUT = IN * 0, in which case the output will be the same type as the input.

It can get more subtle than this. For example, what if you want your output to be a floating point type, and not an integer. The danger of the above approach is if the user passes an integer into the input variable IN, then OUT will also be an integer, undesireably. So usually what I do is: "OUT = IN * 0." Note the extra decimal point, which forces OUT to be at least floating point, but if IN is double precision, then the "0." is safely upcasted to double precision as well.

Beware of using constants like "0.0D" or "0L" in your formulae, because they may end up up-casting your data to a higher data type. Maybe that's okay, but sometimes not. I find myself using code like this:

```
  zero = in(0) * 0
  one  = zero + 1
```

at the beginning of a procedure. That way I can use ZERO and ONE safely in place of 0 and 1 without worry of upcasting IN.

IDL will often drop the final dimension of an array if it is 1. This is often fine, but occasionally it will really put a wrench in the gears. If you really expect an array to have a fixed set of dimensions, then you must be sure to REFORM it. I even do that right after creating arrays!

```
  x = fltarr(nx, ny, nz)  ;; No guarantee NXxNYxNZ if NZ = 1!!!
  x = reform(x, nx, ny, nz, /overwrite)
```

Okay, enough for now!

Good luck,
Craig

--

 -- --------------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------