Subject: Re: declare variables
Posted by Ken Mankoff on Tue, 18 Oct 2005 20:35:09 GMT
View Forum Message <> Reply to Message

On Tue, 18 Oct 2005, qian wrote:
> is it possible I can re-inforce the array to be certain type (or
> let the variables to be certain type throughtout the whole
> program)? So even if I type d=1, it is still a real number?

No. IDL is a very loosly typed language. You can do:

x = 3.14159
x = "sort of pi"

And there exist no mechanism to produce an error, warning, message,
or anything that the variable has changed types.

-k.

Subject: Re: declare variables
Posted by btt on Tue, 18 Oct 2005 21:07:41 GMT
View Forum Message <> Reply to Message

qian wrote:

>
> But if I read in data from other files, in which '1' actually means
> '1.0', is it possible I can re-inforce the array to be certain type (or
> let the variables to be certain type throughtout the whole program)? So
> even if I type d=1, it is still a real number?

Hi,

You can always use the built-in FIX function on the data to recast it as
needed. See the TYPE keyword for fix.

Cheers,
Ben

Subject: Re: declare variables
Posted by James Kuyper on Tue, 18 Oct 2005 21:27:40 GMT
View Forum Message <> Reply to Message

qian wrote:
> Hi,

>
> I am using FORTRAN and IDL at the same time, so sometime I just assume
> some FORTRAN rules when using IDL. I just find out that the variable
> type can change within an IDL program, even you declare it to be some
> specific type. For example:

You don't declare an IDL variable to have a given type. Whenever you
assign a value to an IDL variable, that variable automatically acquires
the data type of whatever it is you assigned into it.

> data=dblarr(2)

This isn't a declaration; it's an assignment statement like any other.
Since dblarr(2) is a 2-element array of doubles, that's the type of
'data'.

> data=[2.3, 3.4]

Since [2.3, 3.4] is a 2-element array of floats, that's the new type of
'data'.

Note: if you type data[0] = 2.3, then 2.3 would have been converted to
a double before storing it in data[0]. A variable's type only changes
when you assign a value to the entire variable, not when you assign to
a single element of an array.

...
> But if I read in data from other files, in which '1' actually means
> '1.0', is it possible I can re-inforce the array to be certain type (or
> let the variables to be certain type throughtout the whole program)? So
> even if I type d=1, it is still a real number?

You don't have to worry about that; it doesn't work that way. When
you're reading data from a text file using commands like:

    READF, infile, myarray

Then myarray must already exist, which means it already has a specific
type, size, and array shape. The interpretatation of the text file is
controlled by the data type and size of myarray. If myarray is
currently an array of 26 integers, the above command will try to
interpret the file as containing 26 integers (and it will complain if
that interpretation doesn't work). If myarray is a 4x4 array of double
precision numbers, it will attempt to interpret the file as containing
16 floating point numbers.

Subject: Re: declare variables
Posted by Mark Hadfield on Tue, 18 Oct 2005 23:37:40 GMT
View Forum Message <> Reply to Message

qian wrote:
> Hi,
>
> I am using FORTRAN and IDL at the same time, so sometime I just assume
> some FORTRAN rules when using IDL. I just find out that the variable
> type can change within an IDL program, even you declare it to be some
> specific type. For example:
>
> data=dblarr(2)
>
> data=[2.3, 3.4]
>
> then now, 'data' is a single precision array.
>
> if I type
>
> data=[2,3]
>
> then it is an integer array.., and data(0)/data(1) = 0 !
>
> I know I should be more careful when programming, always using:
>
> data=[2.3D,4.5D]
>
> But if I read in data from other files, in which '1' actually means
> '1.0', is it possible I can re-inforce the array to be certain type (or
> let the variables to be certain type throughtout the whole program)? So
> even if I type d=1, it is still a real number?

As other respondents have pointed out, IDL is a dynamically typed language so you can't ensure a given variable name is always associated with data of a given type or shape. Still, there are some things you can do to prevent unintended changes of type.

For example, consider the following

```
IDL> data=[2.3,3.4]
IDL> data[*] = [2,3]
IDL> print, data
      2.00000      3.00000
```

The 1st defines data to be a 1-dimensional, 2-element, floating point array. The 2nd assigns new values to the elements of d, but does not change the type or shape. The 3rd confirms that the variable named data is still floating point.

Consider another example

```
IDL> data = fltarr(2)
IDL> read, data
: 2 3
IDL> print, data
      2.00000      3.00000
```

(where the colon indicates that IDL is accepting input from the keyboard--reading from a file has the same result). Here we define data, again as a a 1-dimensional, 2-element, floating point array, then read in values. It doesn't matter that the numbers we read in look like integers--the type and shape of the variable are not altered.

--
Mark Hadfield "Kei puwaha te tai nei, Hoea tahi tatou"
m.hadfield@niwa.co.nz
National Institute for Water and Atmospheric Research (NIWA)
