Subject: Re: Division in a conditional statement
Posted by David Fanning on Sat, 30 Aug 2003 15:46:01 GMT
View Forum Message <> Reply to Message

Hassan Iqbal writes:

> Can anybody tell me what does this mean:
>
> x= a LT total(abs(b))/1e7/NFREE
>
> where:
> b= an array of floating point numbers
> a= a floating point number
> NFREE= an integer
>
> I will be very much thankful. I really need to understand this.

This means that the person who wrote this code
was overly concerned with job security and was
trying to write code only he could decipher.
Either that or he had a sadistic personality, but
I prefer the kinder interpretation. :-)

To interpret this code, you have to have an
understanding of "operator precedence" (you
could look this up on IDL's on-line help).

Parentheses have the highest or first order of precedence
(and your code writer should have used them to
give meaning to this expression). Followed by
multiplication and division operators (fourth
highest order), and followed up by the LT operator,
which has the sixth highest order of pprecedence.

So, everything to the right of the "LT" operator
happens first and a result is stored in a temporary
variable. Then this result is compared to A with the
LT operator. A couple of parentheses would have made
this plain:

```
  x= a LT (total(abs(b))/1e7/NFREE)
```

Operators of the same order of precedence, by the way,
are handled left to right.

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: Division in a conditional statement
Posted by MKatz843 on Mon, 01 Sep 2003 01:49:18 GMT

View Forum Message <> Reply to Message

>> Can anybody tell me what does this mean:
>>
>> x= a LT total(abs(b))/1e7/NFREE
>>
>> where:
>> b= an array of floating point numbers
>> a= a floating point number
>> NFREE= an integer
>>

> x= a LT (total(abs(b))/1e7/NFREE)

In addition to what David wrote, you may have been asking a
very basic question.

The LT (less-than) operator returns true=1 or false=0, and it is
evaluated for each element in the arrays "a" and the temporary array
David put in parantheses above. So x will be an array of 1s and 0s, 1
wherever the less-than statement is true.

Here are a few examples. The behavior is a little different if either
a or b is scalar, or if a and b have multiple elements but their
lengths do not match.

```
IDL> print, [1,2,3,4,5] LT [3,3,3,3,3]
   1   1   0   0   0   ;-- evaluated element-by-element
IDL> print, [1,2,3,4,5] LT [0,0,10,1,2]
   0   0   1   0   0  ;-- similar to the above
IDL> print, [1,2,3,4,5] LT 3
   1   1   0   0   0  ;-- all elements are compared to this scalar
IDL> print, [1,2,3,4,5] LT [5,5]
   1   1  ;-- only the first two elements are compared
IDL> print, [1,2,3,4,5] LT [3,3,3,3,3,3,3,3]
   1   1   0   0   0   ;-- only 5 elements are compared
```

In the last two examples, where the lengths do not match, the result
has the length of the smaller of the two arrays.

Likewise if the 3 in the third example is replaced with [3], a
one-element array, then the result will only have one element as well.

You'll get this same kind of behavior from the other comparison
operators, GT, EQ, LE, and GE.

M. Katz