Subject: Procedures and Functions - Checking Input Data for Validity
Posted by Aram Panasenco on Sat, 17 Apr 2010 03:27:57 GMT
View Forum Message <> Reply to Message

I tried searching IDL Help for a way to simplify checking for validity of passed in arguments, but the only thing I was able to find was documentation of some C macros (please correct me if there actually is something that can easily allow me to validate function/procedure input). So I thought of a way to be able to quickly check if the data falls into a certain type and range.
I propose creating a function that checks input for validity in accordance with a string of rules. You can think of it as an analogue to a FORMAT string. The data and the string both get passed into a function called something along the lines of VALIDATE_INPUT (preferably something shorter). The function returns 1 if the data matches the rules defined by the string, and 0 if it doesn't. Here's what I think the syntax of the string should be:

T

Where:
 - is an optional flag, dictating whether the types specified after it are inclusive (if the flag is not set) or exclusive (if the flag is set)
 T is a list of type codes/names separated by commas. The flag determines whether T specifies what the data should ("inclusive") or shouldn't ("exclusive") be like.
 - is a second optional flag, dictating whether the range(s) specified after it should be exclusive or inclusive
 R is an optional range specification. It can be written in standard mathematical notation (round or square braces)

Type codes/names:
Elements of T can be either type codes from 0 to 15 or type names such as UNDEFINED, DOUBLE, or POINTER. Since it is very easy for a rule processor to differentiate between the two, I think it can be ok to use either one.
All the type codes and names can be found in the documentation for the SIZE function in the IDL reference guide.

Examples of T:
" - NaN"
    All data that contains NaN will return 0
" Double, Float "
    Only Double and Float data will return 1
" - 12,13,14,15"
    All unsigned number types will return 0

Range Notation:
Range defines the numerical boundaries of the data. Range specification will be ignored for non-numerical data. Range will be written in the form (MIN,MAX) or (N), where the braces can be either round or square. A round brace indicates an open boundary (the number is not part of the range), and a square brace indicates a closed boundary (the number is part of the range). The range will accept numbers OR the IDL constant system variables !pi, !dpi, !dtor, and !radeg OR the keyword !infinity (not an actual IDL keyword).

Examples of R:
" (-!pi , !pi) "
    All data within the range, excluding -pi and pi, will return 1
" (0,100] "
    All data within the range, excluding 0, will return 1
" - (0) "
    For single numbers, it won't matter whether you use round or curly brackets.
    In this case, all data that's not equal to 0 will return 1
" "
    All data from -100 to 100, excluding 0, will return 1
" - (0,!infinity)"
    All data from 0 to infinity, excluding 0, will return 0 (all numbers greater than 0 will return 0)
    Note that it's ok, while not recommended (for good math's sake), to brace !infinity with a square bracket

Example of using the function (named VALIDATE_INPUT here):

```
; I have a spherical polygon that I want to
; move around while keeping maximum
; precision. The polygon's spherical
; coordinates are kept in a variable called
; SphVertices, and I don't want the longitude
; values to go out of the range (0,2*!pi) and
; the latitude values to go out of range
; (-!pi,!pi). I can use VALIDATE_INPUT:

pro RenderPolygon, sphVertices
    e = VALIDATE_INPUT(sphVertices , "DOUBLE (0,2*!pi)") and $
          VALIDATE_INPUT(sphVertices[1,*] , "DOUBLE (-!pi,!pi)")
    if (~e) then HandleError("INVALID INPUT: Polygon Vertices")
    ...
    ...
end

; I have a routine that controls a list of files.
; The routine should never be passed a
```

```
; negative number or a non-integer. I can
; use VALIDATE_INPUT:

function SelectFileFromList, index
    e = VALIDATE_INPUT(index , "INT,UINT [0,!infnity)")
    if (~e) then HandleError("INVALID INPUT: Invalid Index for
Filelist")
    ...
    ...
end
```

Please share your thoughts on a function like that. Would it be worthwhile to code something like that. Can you think of any improvements?

~Aram Panasenco
