Subject: Re: CASE statement
Posted by george.mccabe on Wed, 03 Oct 2001 18:49:33 GMT
View Forum Message <> Reply to Message

jeff,

wo are we IDL users and programmers without a logical variable type,
and equivalance, and...

in case anyone wants to use it, i wrote such a function which i use
quite a bit to restrict keyword option values, test for non-zero counts,
among other things. of course, conditional statements are needed
and so i don't call the function where that is undesirable. but it has
its place. (see below)

cheers, george

result Jeff Guerber <jguerber@icesat2.gsfc.nasa.gov>
wrote in message
news:<Pine.GHP.4.32.0109072148460.16214-100000@icesat2.gsfc.nasa.gov>...
> A while back, I tried to think up a function to evaluate the truth of
> its argument, for a general case, and return 0 or 1, for use in situations
> like this. The only thing I could come up with involved a loop over all
> its elements, enclosing an IF (or equivalent a?b:c) statement. OK for
> scalars, not so good for arrays.

```
function oo, var, INVERSE=ton, FAIL_UNDEFINED=udf, HELP=hlp

;+
; PURPOSE:
;   "On or Off", turn any variable into a strict logical type
;   with only two values 0 or 1
;
; INPUTS:
;   var       input string or number.
;   /INVERSE  keyword_flag, "NOT"
;   /FAIL_UNDEFINED
;             fail if variable is undefined instead of returning FALSE
;
; EXAMPLES:
;   if oo("my dog has fleas") then print,"i am not happy"
;
; NOTES:
;
; what happens in IDL when other types are used in a logical context,
;
```

```
; IDL> v=0B    & print,(v),(not v) ;   0 255
; IDL> v=1B    & print,(v),(not v) ;   1 254
; IDL> v=0     & print,(v),(not v) ;       0      -1
; IDL> v=1     & print,(v),(not v) ;       1      -2
; IDL> v=0L    & print,(v),(not v) ;           0         -1
; IDL> v=1L    & print,(v),(not v) ;           1         -2
; IDL> v=0.    & print,(v),(not v) ;      0.00000      1.00000
; IDL> v=1.    & print,(v),(not v) ;      1.00000      0.00000
; IDL> v=0U    & print,(v),(not v) ;       0   65535
; IDL> v=0UL   & print,(v),(not v) ;           0  4294967295
; IDL> v=1U    & print,(v),(not v) ;       1   65534
; IDL> v=1UL   & print,(v),(not v) ;           1  4294967294
; IDL> v=127B  & print,(v),(not v) ; 127 128
; IDL> v=127L  & print,(v),(not v) ;         127        -128
; IDL> v=127   & print,(v),(not v) ;     127    -128
; IDL> v=127.  & print,(v),(not v) ;      127.000      0.00000
; IDL> v=-127. & print,(v),(not v) ;     -127.000      0.00000
; IDL> v=-127  & print,(v),(not v) ;    -127     126
;
; no direct interpretation of strings
; 64 bit numbers act the same as long/float
;
; (refer to IDL Online Help for a description of the NOT operator)
;
; this can lead to unexpected results, eg.
; IDL> if (not 1) then print, "TRUE"
; IDL> if (not 2) then print, "TRUE"
; TRUE
;
; rules applied by this procedure to the interpretation of values of
; all types,
;         Type                  FALSE                TRUE
; string                 NULL or zero length     non-zero length
; byte/uint/ulong/ulong           GT 0                 EQ 0
; integer/long/float/long64       GT 0                 LE 0
; undefined (always FALSE)
; complex/structure/etc. (don't do TRUE/FALSE tests on these types)
;
; note that only a scalar argument is accepted. it's because the
; procedure allows var to be undefined, and it is not possible to create
; an array with undefined elements.
;
; OUTPUTS:
;   v01  function result, type byte restricted to value 0 or 1
;
; HISTORY:
;   09/01  created, ghm
;-
```

```
v01=255B
vfn=-1.
msg0="Error! Logical conversion failed"

USAGE='("SYNTAX: TF = oo( val [, /INVERSE, /FAIL_UNDEFINED] )")'
if n_params() lt 1 or keyword_set(hlp) then begin
  print,format=USAGE
  goto,eject
endif

if (size(var))(0) gt 0 then begin
  message,"Error! Non-scalar argument disallowed",/inform
  goto,eject
endif

vtp=size(var,/type)
vno=where([1,2,3,4,5,12,13,14,15] eq vtp, ano)

case 1 of
  vtp eq 0: begin
    if not keyword_set(udf) then v01=0B
    goto, skip2
    end
  vtp eq 7: begin
    str0=(strlen(var) eq 0)
    if str0 then vfn=0. $
           else vfn=1.
    end
  ano eq 1: begin
    message,/reset
    on_ioerror, JUMPBACK
JUMPBACK:
    if !error_state.code ne 0 then $
      goto, skip2
    afn=float(var)
    on_ioerror,NULL
    if afn gt 0. then vfn=1. $
                 else vfn=0.
    end
  else: begin
    goto, skip2
    end
endcase

if ( vfn ne 0. and vfn ne 1. ) then $
  message,msg0
if keyword_set(ton) then vfn=(not vfn)
```

```
v01=byte(vfn)

skip2:
if ( v01 ne 0B and v01 ne 1B ) then $
  message,msg0
eject:
return, v01
end

-------------------------------------------------
```