Subject: Re: CASE statement
Posted by David Fanning on Fri, 07 Sep 2001 20:23:10 GMT
View Forum Message <> Reply to Message

K Banerjee (kbanerje2@home.com) writes:

```
>  In IDL, is it possible to use mulitple expressions with a single
>  statement?
>
>  Something like:
>
>  PRO CASE1, Tag
>
>    case Tag of
>
>  ;; The following line does not work.
>      ('TIFF') or ('tiff'): begin
>        print, "TAG is TIFF"
>      end
>
>      else: begin
>        print, "TAG is UNKNOWN"
>      end
>
>    endcase
>
>  end
```

No. The proper form of this CASE statement looks
like this:

```
   CASE StrUpCase(tag) OF
 'TIFF': Print, 'Tag is TIFF'
      ELSE  : Print, 'Tag is Unknown'
   ENDCASE
```

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: CASE statement
Posted by Stein Vidar Hagfors H[1] on Fri, 07 Sep 2001 20:39:38 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> writes:

> K Banerjee (kbanerje2@home.com) writes:
>
>> In IDL, is it possible to use mulitple expressions with a single
>> statement?
>>
>> Something like:
>>
>> PRO CASE1, Tag
>>
>> case Tag of
>>
>> ;; The following line does not work.
>>   ('TIFF') or ('tiff'): begin
>>   print, "TAG is TIFF"
>>   end
>>
>>   else: begin
>>   print, "TAG is UNKNOWN"
>>   end
>>
>> endcase
>>
>> end
>
> No. The proper form of this CASE statement looks
> like this:
>
>   CASE StrUpCase(tag) OF
> 'TIFF': Print, 'Tag is TIFF'
>   ELSE : Print, 'Tag is Unknown'
>   ENDCASE

"The proper form" ?

Well, not that I've ever used it myself, but I have seen another nifty form,
something like this:

```
case 1 of
  (tag eq 'TIFF') or $
  (tag eq 'TAFF') or $
  (tag eq 'FIFF') : begin & print,"Tag is TIFF (or a typo thereof)" & end

  (tag eq 'GIF') or $
```

```
   (tag eq 'GUF') : begin & print,"Tag is GIF (or a typo thereof)" & end

   :

   ELSE: print, "Tag is unknown"
end
```

The "ELSE" part can of course be written like this (saves 3 keystrokes :-)

```
   1: print,"Tag is unknown"
```

Whether this is a "proper form", I'll leave up to David :-)

--

 -- --------------
Stein Vidar Hagfors Haugan
ESA SOHO SOC/European Space Agency Science Operations Coordinator for SOHO

NASA Goddard Space Flight Center,   Email: shaugan@esa.nascom.nasa.gov
Mail Code 682.3, Bld. 26, Room G-1,   Tel.: 1-301-286-9028/240-354-6066
Greenbelt, Maryland 20771, USA.   Fax: 1-301-286-0264
 -- --------------

Subject: Re: CASE statement
Posted by David Fanning on Fri, 07 Sep 2001 21:00:12 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan (shaugan@esa.nascom.nasa.gov) writes:

> Whether this is a "proper form", I'll leave up to David :-)

I assure you my former statement is the proper form. This
last form is *WAY* to difficult to explain to anybody. :-)

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: CASE statement

Posted by Pavel A. Romashkin on Fri, 07 Sep 2001 21:30:48 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
>
> Well, not that I've ever used it myself, but I have seen another nifty form,
> something like this:
>
> case 1 of
> (tag eq 'TIFF') or $
> (tag eq 'TAFF') or $
> (tag eq 'FIFF') : begin & print,"Tag is TIFF (or a typo thereof)" & end
... snip

Well, seeing how you can twist even the simpliest syntax, it made me
feel better about the bizarre way I just wrote a 50-code-line Singleton
object. Without ever resorting to Common blocks - are you listening, JD?
:-) I'll post it once I am able to make any object a Singleton type via
a single keyword without exceeding 100 lines of code :-)

cheers,
Pavel

P.S. Let's just say (c) that inability to make progress with what you
are supposed to be doing makes you creative in things you are not
supposed to be doing :-(

Subject: Re: CASE statement
Posted by Brian Jackel on Fri, 07 Sep 2001 21:41:35 GMT
View Forum Message <> Reply to Message

Well, you could try this:

```
CASE (1) OF
(this OR that): x=1
(something AND somethingelse): x=2
(NOT theotherthing): x=3
ELSE:MESSAGE,'Error- no match found for case statement'
ENDCASE
```

Kinda ugly, but gets the job done.

Brian

K Banerjee wrote:
>
> In IDL, is it possible to use mulitple expressions with a single

```
>  statement?
>
>  Something like:
>
>  PRO CASE1, Tag
>
>    case Tag of
>
>  ;; The following line does not work.
>      ('TIFF') or ('tiff'): begin
>        print, "TAG is TIFF"
>      end
>
>      else: begin
>        print, "TAG is UNKNOWN"
>      end
>
>    endcase
```

Subject: Re: CASE statement
Posted by Kristian Kjaer on Fri, 07 Sep 2001 22:17:31 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
...
> case 1 of
...

The following reads A LOT better, I think:

```
TRUE=1
...
case TRUE of
...
```

Lets just say: What's in a name? Everything!

- Kristian Kjær, Risø Natl. Lab., Denmark

Subject: Re: CASE statement
Posted by Pavel A. Romashkin on Fri, 07 Sep 2001 23:16:54 GMT
View Forum Message <> Reply to Message

Kristian Kjaer wrote:

>
> The following reads A LOT better, I think:
>
> TRUE=1
> ...
> case TRUE of

Now, this is pretty scary! This syntax implies that its ok to have
"True" OF, lets say, "Almost_true", "Sometimes_true", "False" etc...
I'd better stick to David's format :-)
cheers,
Pavel

Subject: Re: CASE statement
Posted by gmccabe14 on Sat, 08 Sep 2001 01:44:30 GMT
View Forum Message <> Reply to Message

hi,

"the square root of 3 is 2, for big enough values of 3."

i like case 1 of. my thanks to the users for another creative solution.

george
george.mccabe@gsfc.nasa.gov

Brian Jackel <bjackel@phys.ucalgary.ca> wrote in message
news:<3B993F0F.8DBF705A@phys.ucalgary.ca>...
> Well, you could try this:
>
> CASE (1) OF
> (this OR that): x=1
> (something AND somethingelse): x=2
> (NOT theotherthing): x=3
> ELSE:MESSAGE,'Error- no match found for case statement'
> ENDCASE
>
> Kinda ugly, but gets the job done.
>
> Brian
>
> K Banerjee wrote:
>>
>> In IDL, is it possible to use mulitple expressions with a single
>> statement?
>>

```
>>  Something like:
>>
>>  PRO CASE1, Tag
>>
>>    case Tag of
>>
>>  ;; The following line does not work.
>>      ('TIFF') or ('tiff'): begin
>>        print, "TAG is TIFF"
>>      end
>>
>>      else: begin
>>        print, "TAG is UNKNOWN"
>>      end
>>
>>    endcase
```

Subject: Re: CASE statement
Posted by Jeff Guerber on Sat, 08 Sep 2001 03:18:15 GMT
View Forum Message <> Reply to Message

On Fri, 7 Sep 2001, Brian Jackel wrote:

```
>   CASE (1) OF
>   (this OR that): x=1
>   (something AND somethingelse): x=2
>   (NOT theotherthing): x=3
```

Careful!!! Notice:

```
IDL> print, not 0, not 0b, not 1, not 1b
     -1 255      -2 254
IDL> print, (not (1 eq 2)) eq (1 ne 2)
  0
IDL>
```

Unfortunately, AND, OR, XOR, and NOT all operate _bitwise_, at least on integral types. The first two selectors in your CASE should be OK if the operands are all expressions that evaluate to 0 or 1 (which the relationals EQ, NE, GT, GE, LT, and LE all do; but watch out for the general case!). The third one very likely won't do what you intend.

This is why I've argued a couple times for a true logical type, like Fortran's, with "true" and "false" system constants and Boolean operators that always return them. More recently, though, I discovered some examples in the manual that use the 0 or 1 returned from the relationals in arithmetic expressions, so I guess we're stuck with them; I'd be mostly

satisfied with operators that return only 0 or 1, and constants !true=1 and !false=0. (Hmmmm. Perhaps a type that can _only_ have the values 0 or 1, with operand conversion based on truth value??)

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.

> ELSE:MESSAGE,'Error- no match found for case statement'
> ENDCASE
>
> Kinda ugly, but gets the job done.

Agreed, and I use the idiom myself; but, you've got to be more careful with it than I for one think you should have to be!

Jeff Guerber
Raytheon ITSS
NASA Goddard Space Flight Ctr
Oceans & Ice Branch

P.S. I've long thought that a language where comparisons can be distributed, as in English, would be very handy:

if expression eq A or B or C or D then...

Subject: Re: CASE statement
Posted by Martin Downing on Mon, 10 Sep 2001 11:45:43 GMT
View Forum Message <> Reply to Message

>>> In IDL, is it possible to use mulitple expressions with a single
>>> statement?
>>>
>>> Something like:
>>>
>>> PRO CASE1, Tag
>>>
>>> case Tag of
>>>
>>> ;; The following line does not work.
>>> ('TIFF') or ('tiff'): begin
>>> print, "TAG is TIFF"
>>> end
[snip]
> Well, not that I've ever used it myself, but I have seen another nifty

form,
> something like this:
>
> case 1 of
> (tag eq 'TIFF') or $
> (tag eq 'TAFF') or $
> (tag eq 'FIFF') : begin & print,"Tag is TIFF (or a typo thereof)" & end
>
[snip]
>
> Whether this is a "proper form", I'll leave up to David :-)
>

It seems to me you all have a SWITCH-phobia :-)
have i missed something or shouldnt we just be writing:

```
SWITCH tag of
  "TIFF":
   "TAFF": BEGIN
        print, "tagged image format"
        BREAK
    END
    ELSE: print, "unknown format"
ENDSWITCH
```

Also, please dont encorage RSI to produce a !TRUE constant, as I and other C/C++ users have trouble not reading this as "NOT TRUE". I guess it would be OK to have !VALUES.TRUE though.

cheers
Martin

Martin Downing,
Clinical Research Physicist,
Orthopaedic RSA Research Centre,
Woodend Hospital, Aberdeen, AB15 6LS.

Subject: Re: CASE statement
Posted by R.Bauer on Mon, 10 Sep 2001 11:54:51 GMT
View Forum Message <> Reply to Message

Martin Downing wrote:
>
>>>> In IDL, is it possible to use mulitple expressions with a single
>>>> statement?
>>>>

>>>> Something like:
>>>>
>>>> PRO CASE1, Tag
>>>>
>>>>   case Tag of
>>>>
>>>> ;; The following line does not work.
>>>>   ('TIFF') or ('tiff'): begin
>>>>   print, "TAG is TIFF"
>>>>   end
> [snip]
>> Well, not that I've ever used it myself, but I have seen another nifty
> form,
>> something like this:
>>
>> case 1 of
>>   (tag eq 'TIFF') or $
>>   (tag eq 'TAFF') or $
>>   (tag eq 'FIFF') : begin & print,"Tag is TIFF (or a typo thereof)" & end
>>
> [snip]
>>
>> Whether this is a "proper form", I'll leave up to David :-)
>>
>
> It seems to me you all have a SWITCH-phobia :-)

No I am not I just waiting.
But I am missing strupcase(tag) ...

cheers

Reimar

> have i missed something or shouldnt we just be writing:
>
> SWITCH tag of
>   "TIFF":
>   "TAFF": BEGIN
>   print, "tagged image format"
>   BREAK
>   END
>   ELSE: print, "unknown format"
> ENDSWITCH
>
> Also, please dont encorage RSI to produce a !TRUE constant, as I and other
> C/C++ users have trouble not reading this as "NOT TRUE". I guess it would be

> OK to have !VALUES.TRUE though.
>
> cheers
> Martin
>
> ---
> Martin Downing,
> Clinical Research Physicist,
> Orthopaedic RSA Research Centre,
> Woodend Hospital, Aberdeen, AB15 6LS.

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-1)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg1/
========== ======
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg1/idl_icglib/idl_lib_intro.h tml

http://www.fz-juelich.de/zb/text/publikation/juel3786.html
========== ======

read something about linux / windows
http://www.suse.de/de/news/hotnews/MS.html

Subject: Re: CASE statement
Posted by marc schellens[1] on Thu, 13 Sep 2001 06:01:31 GMT
View Forum Message <> Reply to Message

Stein Vidar Hagfors Haugan wrote:
>
> "The proper form" ?
>
> Well, not that I've ever used it myself, but I have seen another nifty form,
> something like this:
>
> case 1 of
> (tag eq 'TIFF') or $
> (tag eq 'TAFF') or $
> (tag eq 'FIFF') : begin & print,"Tag is TIFF (or a typo thereof)" & end
>
> (tag eq 'GIF') or $
> (tag eq 'GUF') : begin & print,"Tag is GIF (or a typo thereof)" & end
>

>   :
>
>   ELSE: print, "Tag is unknown"
> end
>
> The "ELSE" part can of course be written like this (saves 3 keystrokes :-)
>
>   1: print,"Tag is unknown"
>
> Whether this is a "proper form", I'll leave up to David :-)
>

I think thats why for most people the term IDL-users is more
appropriate than IDL-programmers...

:-) marc

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

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