
Subject: Re: How to delete a variable

Posted by [jigga1206](#) on Fri, 18 Apr 2014 20:29:59 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi Wayne,

I'm not expert enough (or have IDL 8) to test if this will work for your problem, but here's a program from ITT called Vardel that may work in some limited scenarios. It accepts multiple variables at once and can be called from where ever is necessary, but pointers may not work.

Take care

```
*****
;
;+
; <PRO VARDEL>
; The purpose of this program is to provide an alternative to IDL's internal DELVAR
; routine that provides support for wildcards similar to a standard UNIX shell.
; This program also supports designation of variable names to delete as an array of
; strings, and can be executed within programs, unlike DELVAR. Variables at the
; $MAIN$ level will continue to be listed as <Undefined> within the main program's
; system table.<BR>
; <BR>
; User-Contributed Library webpage :
; <A
HREF="http://www.rsinc.com/codebank/search.asp?FID=398">http://www.rsinc.com/codebank/se
arch.asp?FID=398</A>
;
;
; @Param
; matchStrings {in} {required} {type=string}
; A scalar string or array of strings that is used to identify variables to be deleted.
;
;
; @Author
; Adam O'Connor<BR>
; <A HREF="mailto:adam_e_oconnor@yahoo.com">&lt;adam_e_oconnor@yahoo.com&gt;
</A>
;
;
; @Copyright
; Copyright &#169; 2006 ITT Industries. All rights reserved.
;
;
; @Categories
; Language, Variable, Delete, Deletion, Memory, Scope, Programming
;
;
; @Examples
; <PRE>
; - Start a new session of IDL and create some variables :
;
;
; IDL Version 6.2, Microsoft Windows (Win32 x86 m32).
; IDL> a1=indgen(10)
```

```

; IDL> a2=indgen(10)
; IDL> b1=indgen(10)
; IDL> b2=indgen(10)
; IDL> c1=indgen(10)
; IDL> c2=indgen(10)
; IDL> deer='deer'
; IDL> door='door'
;
; - Execute the HELP routine to get information on the currently defined variables within
;   the IDL session :
;
; IDL> help
; % At $MAIN$
; A1      INT      = Array[10]
; A2      INT      = Array[10]
; B1      INT      = Array[10]
; B2      INT      = Array[10]
; C1      INT      = Array[10]
; C2      INT      = Array[10]
; DEER    STRING   = 'deer'
; DOOR    STRING   = 'door'
;
; - Execute VARDEL procedure and delete all variables that start with the letter "b" :
;
; IDL> vardel, 'b*'
;
; - Execute the HELP routine to get information on the currently defined variables within
;   the IDL session :
;
; IDL> help
; % At $MAIN$
; A1      INT      = Array[10]
; A2      INT      = Array[10]
; B1      UNDEFINED = <Undefined>
; B2      UNDEFINED = <Undefined>
; C1      INT      = Array[10]
; C2      INT      = Array[10]
; DEER    STRING   = 'deer'
; DOOR    STRING   = 'door'
;
; - The "?" wildcard is also supported and the argument for VARDEL can be an array of strings :
;
; IDL> vardel, ['d??r', 'a*']
; IDL> help
; % At $MAIN$
; A1      UNDEFINED = <Undefined>
; A2      UNDEFINED = <Undefined>
; B1      UNDEFINED = <Undefined>

```

```

; B2      UNDEFINED = <Undefined>
; C1      INT      = Array[10]
; C2      INT      = Array[10]
; DEER    UNDEFINED = <Undefined>
; DOOR    UNDEFINED = <Undefined>
;
; - Although the deleted variables are still listed in the current session, the SAVE
;   procedure will not save them to the *.sav file output to disk :
;
; IDL> save, /variables, file='data.sav'
;
; - When you restore this *.sav file back into a new session of IDL, only the defined
;   variables are retained :
;
; IDL Version 6.2, Microsoft Windows (Win32 x86 m32).
; IDL> restore,'data.sav'
; IDL> help
; At $MAIN$
; C1      INT      = Array[10]
; C2      INT      = Array[10]
;
; - The VARDEL routine can also be executed within programs. Consider for example a
;   procedure named "vardel_test" :
;
; PRO vardel_test
; a1=indgen(10)
; a2=indgen(10)
; help,a1,a2
; vardel, 'a*'
; help,a1,a2
; END
;
; - Executing this "vardel_test" program results in the following output :
;
; IDL> vardel_test
; A1      INT      = Array[10]
; A2      INT      = Array[10]
; A1      UNDEFINED = <Undefined>
; A2      UNDEFINED = <Undefined>
; </PRE>
;
; @History
;   Created: February, 2006
;
; @Requires
;   6.1
;
; @Restrictions

```

```

; Although this program was generated by an ITT employee, it is in no way supported,
; maintained, warranted, or guaranteed by ITT Industries. Consequently, the ITT
; Technical Support Group is not obligated to address questions related to this
; program. Use this program at your own risk.
;-
PRO VARDEL, matchStrings
COMPILE_OPT idl2
;error handling ... reset !ERROR_STATE:
MESSAGE, /RESET
;catch error:
CATCH, error
IF (error NE 0) THEN BEGIN ;error caught:
    ;display error:
    MESSAGE, !ERROR_STATE.MSG, /CONTIN
    ;terminate execution of this program:
    RETURN
ENDIF
;ignore beta and development build versions of IDL because string to float conversion will fail:
betaTest = STRPOS (STRLOWCASE (!VERSION.RELEASE), 'beta')
buildTest = STRPOS (STRLOWCASE (!VERSION.RELEASE), 'build')
;check to make sure the version of IDL running is 6.1 or newer:
if (betaTest EQ -1) and (buildTest EQ -1) then begin
    if (FLOAT (!VERSION.RELEASE) LT 6.1) then begin
        MESSAGE, 'Routine is only supported in IDL version 6.1 or newer.', /CONTIN
        RETURN
    endif
endif
;check to make sure program was called with the appropriate number of arguments:
IF (N_PARAMS () NE 1) THEN BEGIN
    MESSAGE, 'Incorrect number of arguments.', /CONTIN
    RETURN
ENDIF
;check to make sure that argument is a string:
IF (SIZE (matchStrings, /TYPE) NE 7) THEN BEGIN
    MESSAGE, 'Argument must be string data type.', /CONTIN
    RETURN
ENDIF
;get variable names at previous interpreter frame level:
varNames = SCOPE_VARNAME (COUNT=varCount, LEVEL=-1)
;if no variables are found then return:
IF (varCount EQ 0) THEN RETURN
;loop through the match strings:
FOR i=0, N_ELEMENTS(matchStrings)-1 DO BEGIN
    ;use STRMATCH to perform string matching with wildcard support:
    match = STRMATCH (varNames, matchStrings[i], /FOLD_CASE)
    ;find the names of the variables that match:
    matchIndex = WHERE (match EQ 1, matchCount)
    ;if no matching variables are found then return:

```

```

IF (matchCount EQ 0) THEN CONTINUE
;loop through the matches:
FOR j=0, matchCount-1 DO BEGIN
    ;extract variable name:
    variableName = varNames [ matchIndex[j] ]
    ;make sure variable is not already <Undefined>:
    IF (SIZE ( (SCOPE_VARFETCH (variableName, LEVEL=-1) ), /TYPE) EQ 0) THEN
CONTINUE
    ;act upon variable using HELP procedure and wrap with TEMPORARY function in order to
delete:
    HELP, (TEMPORARY ( (SCOPE_VARFETCH (variableName, LEVEL=-1) ) ), OUTPUT=void
    ENDFOR
ENDFOR
END
;*****
,

```
