Subject: *** HERE IT IS: arrex, version 1.0 ***
Posted by Martin Schultz on Thu, 20 May 1999 07:00:00 GMT
View Forum Message <> Reply to Message

Hi folks,

  couldn't let go .. Please find a first version of ARREX.PRO attached.
I encourage you to
do a lot of testing and tell me what doesn't work as you expect. This
version is limited to a maximum of 4 dimensions, but once it is stable,
it is very easy to extend it. Also, the internal function
arrex_ComputeInd contains practically everything one needs to write an
ARREXI function to return the indices instead of the values. I included
some debug output for now to make it clear that his is a test version.
Haven't done any timing tests, but a test with
    a = findgen(1000,100,100)
    print,arrex(a,-1,-1,[100,10,10])
runs reasonably fast.

Enjoy!
Martin.

|||||||||||||||\\\\\\\\\\\\\\\----------////////////// //|||||||||||||||
Martin Schultz, DEAS, Harvard University, 29 Oxford St., Pierce 109,
Cambridge, MA 02138   phone (617) 496 8318   fax (617) 495 4551
e-mail mgs@io.harvard.edu   web http://www-as/people/staff/mgs/
```
 ;------------------------------------------------------------ --
; $Id$
;+
; NAME:
;       ARREX  (function)
;
; PURPOSE:
;       This function extracts a multi-dimensional subset
;       of an array. For each dimension the start index,
;       end index, and stride can be specified - default values
;       are 0 (i.e. first start index), "all" (i.e. last end
;       index), and 1 (i.e. every element). A negative stride
;       causes reversion of the respective array dimension.
;
; CATEGORY:
;       Math
;
; CALLING SEQUENCE:
;       Result = ARREX(Array [, Starti [,Endi [, Stride ]]] [,/Reform])
;
; INPUTS:
```

```
;       ARRAY -> The parent array from which the extraction shall be made.
;          Currently up to 4 dimensions are supported.
;
;       STARTI -> An array with starting indices. Each element of STARTI
;          corresponds to one dimension of ARRAY. If STARTI has less
;          elements than ARRAY dimensions, the remainder is assumed 0.
;
;       ENDI   -> An array with ending indices. Each element of ENDI
;          corresponds to one dimension of ARRAY. If ENDI has less
;          elements than ARRAY dimensions, the remainder is set to the
;          maximum possible value for that dimension.
;
;       STRIDE -> An array with stride values. A stride of 1 (default)
;          signifies extraction of every element, 2 returns every
;          second element for that dimension, etc. Negative values
;          cause the respective dimension to be reversed.
;          Each value of STRIDE corresponds to one dimension of ARRAY.
;          If STRIDE has less elements than ARRAY dimensions, the
;          remainder is assumed 1.
;
; KEYWORD PARAMETERS:
;       /REFORM -> Set this keyword to eliminate unnecessary dimensions
;          in the result.
;
; OUTPUTS:
;       A subset of the original array. This will have the same
;       dimensionality as ARRAY unless the REFORM keyword is set.
;
; SUBROUTINES:
;       Function arrex_ComputeInd
;
; REQUIREMENTS:
;       Uses the reverse function from the IDL library
;
; NOTES:
;       Created after discussion in newsgroup idl-pvwave.
;
;       This version contains some debug output.
;
; EXAMPLE:
;       A = indgen(10,10)
;       print,arrex(A,[2,1],-1,[2,4])
;       ; yields  12  14  16  18
;       ;         52  54  56  58
;       ;         92  94  96  98
;       print,arrex(A,[10,1],[1,10],[5,5])
;       ; yields  19  15
;       ;         69  65
```

```
;       ; note that stride for dimension 1 is adjusted automatically.
;
; MODIFICATION HISTORY:
;       mgs, 20 May 1999: VERSION 1.00
;           - with a lot of input from L. Gumley, S. Vidar, and others
;
;-
; Copyright (C) 1999, Martin Schultz, Harvard University
; This software is provided as is without any warranty
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine arrex"
 ;--------------------------------------------------------- --


function arrex_ComputeInd,DStart,DEnd,DStride

    ; get number of elements to be returned for current dimension
    ; Integer division!
    NEl = ( ( (DEnd>DStart) - (DStart<DEnd) ) / DStride ) + 1

    ; Compute index array
    if (DStart le DEnd) then $
      Ind = lindgen(NEl)*DStride + DStart  $
    else $
      Ind = DStart - lindgen(NEl)*DStride

    return,Ind
end



function arrex,Array,Starti,Endi,Stride,Reform=DoReform

   ; return quietly if array is undefined or a scalar
   if (n_elements(Array) eq 0) then $
      return,0L
   if (n_elements(Array) eq 1) then $
      return,Array


   ; get array dimensions
   NDims = size( Array, /N_Dimensions )
   Dims = size( Array, /Dimensions )
```

```
; ### DEBUG
help,Array

    ; check parameters and set defaults
    ; if parameter is not present use
    ;    0 as default for Starti
    ;    Dim[i] as default for Endi
    ;    1 as default for Stride
    ; if parameter has less dimensins than Array fill
    ; remainder with defaults
    ; if Starti or Endi are negative (-1) use defaults
    ; if Starti > Endi, Stride must be < 0. This reverses
    ; the respective dimension.

    DStart = lonarr(NDims)   ; array with 0
    DEnd   = Dims - 1L       ; last elements
    DStride = lonarr(NDims) + 1L  ; every element

    if (n_elements(Starti) gt 0) then begin
      if (n_elements(Starti) gt NDims) then begin
        message,'START contains more numbers than ARRAY dimensions!', $
            /Continue
        return, Array
      endif
      ; since default is 0 anyway, only copy values in Starti that
      ; are greater zero
      ind = where(Starti gt 0L)
      if (ind[0] ge 0) then $
        DStart[ind] = ( Starti[ind] < Dims[ind] )
    endif
; ### DEBUG
print,'Start indices: ',fix(DStart)


    if (n_elements(Endi) gt 0) then begin
      if (n_elements(Endi) gt NDims) then begin
        message,'END contains more numbers than ARRAY dimensions!', $
            /Continue
        return, Array
      endif
      ; Only copy values in Endi that are at least zero
      ind = where(Endi ge 0L)
      if (ind[0] ge 0) then $
        DEnd[ind] = ( Endi[ind] < Dims[ind] )
    endif
; ### DEBUG
print,'End indices: ',fix(DEnd)
```

```
    if (n_elements(Stride) gt 0) then begin
      if (n_elements(Stride) gt NDims) then begin
        message,'STRIDE contains more numbers than ARRAY dimensions!', $
            /Continue
        return, Array
      endif
      ; Only copy values in Stride that are not zero
      ind = where(Stride ne 0L)
      if (ind[0] ge 0) then $
        DStride[ind] = Stride[ind]
      ; Check if Starti, Endi, and Stride are consistent
      ; Starti < Endi with Stride < 0 leads to "collapse" of array
      for i=0,NDims-1 do begin
        if (DStart[i] lt DEnd[i] AND DStride[i] lt 0) then begin
          message,'Invalid parameters for dimension '+strtrim(i+1,2) + $
              '! Adjusting STRIDE.',/Continue
          DStride[i] = -DStride[i]
        endif
        if (DStart[i] gt DEnd[i] AND DStride[i] gt 0) then begin
          message,'Invalid parameters for dimension '+strtrim(i+1,2) + $
              '! Adjusting STRIDE.',/Continue
          DStride[i] = -DStride[i]
        endif
      endfor
    endif else begin   ; change default to include negative strides
                       ; where ENDI lt STARTI
      ind = where(DEnd lt DStart)
      if (ind[0] ge 0) then $
        DStride[ind] = -1L
    endelse

; ### DEBUG
print,'Stride: ',fix(DStride)

    ; For extraction, change Stride to all positive
    ; Save info which dimensions shall be reversed (just for debugging)
    NeedReverse = intarr(NDims)
    ind = where(DStride lt 0)
    if (ind[0] ge 0) then $
      NeedReverse[ind] = 1

    DStride = abs(DStride)


    ; ****** actual extraction here  ******
    Result = Array
```

```
case (NDims) of

  1 : begin
        i = 0
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[ind])
      endif

  2 : begin
        i = 0
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[ind,*])
        i = 1
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[*,ind])
      endif

  3 : begin
        i = 0
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[ind,*,*])
        i = 1
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[*,ind,*])
        i = 2
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[*,*,ind])
      endif

  4 : begin
        i = 0
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[ind,*,*,*])
        i = 1
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[*,ind,*,*])
        i = 2
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[*,*,ind,*])
        i = 3
        Ind = arrex_ComputeInd(DStart[i],DEnd[i],DStride[i])
        Result = temporary(Result[*,*,*,ind])
      endif

  else : message,'TEST version. Not more than 4 dimensions allowed.', $
           /Continue
endcase
```

```
    ; Apply reform if requested
    if (keyword_set(DoReform)) then $
       Result = reform(temporary(Result))


    return,Result

end
```

File Attachments

1) arrex.pro, downloaded 76 times