Subject: Re: Sort
Posted by Martin Schultz on Wed, 05 Nov 1997 08:00:00 GMT
View Forum Message <> Reply to Message

This is a multi-part message in MIME format.

----------167E2781446B
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Neil Winrow wrote:
>
> Can anybody please offer their help!
>
> I have three sets of data, X, Y, Z. I read these sets of data into one
> big array. I would like to sort one of the sets of data within the big
> array, lets say sort the Z data. Is it then possible for the
> corresponding values in the X and Y to also sort to the correct values.
> Hope someone can understand, and offer me a solution.
>
> Many Thanks In Advance
>
> Neil Winrow.

You can find a routine that does this in my EXPLORE package which you
find on the web page given below. The relevant procedure is
multisort.pro, and there is a widget "interface" for it in w_sort.pro.
I just checked trough these again, and found that w_sort actually
contains a copy of multisort (oh yes, I definitively have to clean
this EXPLORE package...). I'll attach w_sort.pro to this message,
feel free to ask for further help.

Regards,
Martin.

--
-- -------
Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
186 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
IDL-homepage: http://www-as.harvard.edu/people/staff/mgs/idl/
-- -------

----------167E2781446B
Content-Type: text/plain; charset=us-ascii; name="w_sort.pro"
Content-Transfer-Encoding: 7bit
Content-Disposition: inline; filename="w_sort.pro"

```
; $Id: w_calc.pro,v 1.0 1997/05/23 mgs $

;+
; NAME:
;     w_calc
; PURPOSE:
;     creates a widget that defines a sort order for a data set
;
; CATEGORY:
; Modal widgets.
;
; CALLING SEQUENCE:
; widget = w_sort(parent)
;
; INPUTS:
;       PARENT - The ID of the parent widget.
;
; KEYWORD PARAMETERS:
; UVALUE - Supplies the user value for the widget.
;     VALUE  - Supplies the initial entries of the list (string array)
;
; OUTPUTS:
;       The ID of the created widget is returned.
;       And an event info field that contains necessary information on
;       the requested sort procedure (sort indices and reverse's)
;
; COMMON BLOCKS:
; None.
;
; SIDE EFFECTS:
;
; PROCEDURE:
;
; MODIFICATION HISTORY:
;-



 ;------------------------------------------------------------ ------------------


; ======  here are the routines that actually do thew work ======
```

```
pro multisort,data,index=index,revert=revert,level=level

; sorts a 2-dim data array (vars, obs) recursively for all indices passed
; in index. If no index field is passed, the data is sorted according to
; the first variable (index = 0)
; /revert produces a data set with reversed sort order in all indices
; the level keyword is for internal purposes only


if(n_elements(index) lt 1) then index = 0
if(not keyword_set(level)) then level=0
; handle revert parameter
if(n_elements(revert) eq 1 AND keyword_set(revert)) then revertall=1  $
else revertall = 0
if(n_elements(revert) gt 1 AND revert(0)) then revertthis=1 else revertthis=0
if(n_elements(revert) gt 1) then reverti = 1 else reverti=0

; due to pecularities of the implementation, all revert indices following
; a set value must be reverted
if (reverti and level eq 0) then begin
  for i = 0,n_elements(revert)-2 do begin
     if(revert(i)) then for j=i+1,n_elements(revert)-1 do $
        if(revert(j)) then revert(j)=0 else revert(j) = 1
  endfor
endif


; perform simple sort
nind = n_elements(index)
x = data(index(0),*)
ind = sort(x)
data = data(*,ind)

; extract boundaries of equal values in major index
uind = uniq(x(ind))
uind = [ 0, uind ]   ; add 0 as first startindex

; create subset of index terms for recursive sort
; if only one index was passed, the sort process is terminating
if nind gt 1 then begin
   subindex = index(1:nind-1)
   if (reverti) then subrevert = revert(1:nind-1) else subrevert=0
   if(n_elements(subrevert) eq 1) then subrevert = subrevert(0)
endif else goto,revertdata

; perform sort on subsets of data with equal major variable
```

```
for i=0,n_elements(uind)-2 do begin
   i1 = uind(i)+1
   i2 = uind(i+1)
   if(i2 ge i1) then begin
      subdat = data(*,i1:i2)
       multisort,subdat,index=subindex,revert=subrevert,level=level +1
   endif else subdat = data(*,uind(i))
   if (i eq 0) then newdat = transpose(subdat)  $
   else newdat = [ newdat, transpose(subdat) ]
endfor

data = transpose(newdat)

; reverse data if positive sort completed and keyword revert set
revertdata:
; if(level eq 0 AND revertall) then data = reverse(data,2) $
; else if(revertthis) then data = reverse(data,2)
if(revertall OR revertthis) then data = reverse(data,2)

return
end


pro handle_sort,data,info

ind = where(info.index ge 0,count)
if (count le 0) then return  ; no valid selection

index = info.index(ind)
revert = info.revert(ind)

multisort,data,index=index,revert=revert

return
end



pro sorttest,index=index,revert=revert

if(not keyword_set(index)) then index = [1,2]
if(not keyword_set(revert)) then revert=0

col0 = findgen(20)
col1 = [ 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4 ]
col2 = [ 2, 2, 2, 3, 5, 4, 3, 2, 2, 2, 4, 3, 3, 3, 5, 4, 4, 3, 3, 1 ]
col3 = [ 8, 7, 6, 5, 4, 3, 2, 1, 8, 7, 6, 5, 4, 3, 2, 1, 8, 7, 6, 5 ]
```

```
data = [  transpose(col0),transpose(col1),transpose(col2),transpose(co l3) ]

multisort,data,index=index,revert=revert

for i=0,n_elements(data(0,*))-1 do $
   print,fix(data(0,i)),data(1,i),data(2,i),data(3,i)

print

return
end


; ===== and here comes the widgety stuff =====

FUNCTION xsort_event, event

returnval = 0     ; default: behave like a procedure

; get state
stateholder = widget_info(event.handler,/child)
widget_control,stateholder,get_uvalue=state,/no_copy


if (event.id eq state.ids(0)) then begin  ; button pressed (OK or cancel)
  if(event.value eq 0) then begin
    ids = state.ids
    ltype = state.ltype
    if(ltype) then begin
      x1 = widget_info(ids(1),/list_select)-1
      x2 = widget_info(ids(2),/list_select)-1
      x3 = widget_info(ids(3),/list_select)-1
    endif else begin
      x1 = widget_info(ids(1),/droplist_select)-1
      x2 = widget_info(ids(2),/droplist_select)-1
      x3 = widget_info(ids(3),/droplist_select)-1
    endelse
    widget_control,ids(4),get_value=r1
    widget_control,ids(5),get_value=r2
    widget_control,ids(6),get_value=r3

    index = [x1, x2, x3]
    revert = [r1, r2, r3 ]
    result = { index:index, revert:revert }
  endif else result = 0

  returnval = { id:0L, top:event.top, $
              handler:0L, value:1-event.value, info:result }
```

```
endif

; restore state
widget_control,stateholder,set_uvalue=state,/no_copy

return,returnval
end

 ;------------------------------------------------------------ ------------------


FUNCTION w_sort, UVALUE=uval, SPECIES=species

; pass a species list that provides the possible arguments



  ON_ERROR, 2     ;return to caller

 ; Defaults for keywords
  IF NOT (KEYWORD_SET(uval))  THEN uval = 0
  if (not keyword_set(species)) then title = ''

sstr = species
; decide whether to use droplist or list for species
if (n_elements(sstr) gt 25) then ltype=1 else ltype=0
sstr = [' ', sstr]

base = widget_base(/column,uvalue=uval,event_func="xsort_event",  $
                title='SORT')

if(ltype) then lower = widget_base(base,/row)  $
else lower = widget_base(base,/row,ysize=80)

dumbase = widget_base(lower,/column)
dum = widget_label(dumbase,value = 'PRIMARY KEY')
if(ltype) then begin
   x1 = widget_list(dumbase,value = sstr,ysize=9)
   widget_control,x1,set_list_select=1
endif else begin
   x1 = widget_droplist(dumbase,value = sstr)
   widget_control,x1,set_droplist_select=1
endelse
r1 = cw_bgroup(dumbase,/nonexclusive,['reverse'])

dumbase = widget_base(lower,/column)
dum = widget_label(dumbase,value = 'SECONDARY KEY')
if(ltype) then begin
```

```
   x2 = widget_list(dumbase,value = sstr,ysize=9)
   widget_control,x1,set_list_select=1
endif else begin
   x2 = widget_droplist(dumbase,value = sstr)
   widget_control,x1,set_droplist_select=1
endelse
r2 = cw_bgroup(dumbase,/nonexclusive,['reverse'])

dumbase = widget_base(lower,/column)
dum = widget_label(dumbase,value = 'TERTIARY KEY')
if(ltype) then begin
   x3 = widget_list(dumbase,value = sstr,ysize=9)
   widget_control,x1,set_list_select=1
endif else begin
   x3 = widget_droplist(dumbase,value = sstr)
   widget_control,x1,set_droplist_select=1
endelse
r3 = cw_bgroup(dumbase,/nonexclusive,['reverse'])


; OK and Cancel buttons
but = cw_bgroup(base,/row,['  OK  ','Cancel'],space=10)

; set ids in state structure
ids = [ but, x1, x2, x3, r1, r2, r3 ]
state = { ltype:ltype, ids:ids }
 widget_control,widget_info(base,/child),set_uvalue=state,/no _copy

return,base
end
```

----------167E2781446B--