```
REM BBC BASIC FOR WINDOWS (BB4W) program to perform spatial filtering
REM on a Windows BMP RGB file.

REM (c) Alan Roberts 2010

SYS "SetWindowText", @hwnd%, "(5) Generate filtered copy of RGB BMP file."

REM first get the filter coefficients

filter% = OPENIN "Noise filter.txt"

IF filter% = 0 THEN
  PRINT " Can't find noise filter file (Noise filter.txt). Press any key to exit."
  IF GET QUIT                                 : REM we're not doing any more, so close the window
ENDIF

line$ = FNinput(filter%)                      : REM read the first line from the file
IF line$ <> "Noise filter" THEN
  PRINT " File 'Noise filter.txt' is not correct. Press any key to exit."
  IF GET QUIT                :REM we're not doing any more, so close the window
ENDIF

REPEAT                                        : REM scan the file, ignoring comments (lines starting with //), looking for the filter
  line$ = FNinput(filter%)
UNTIL INSTR(line$, "Filter-") = 1 OR EOF# filter%
IF EOF# filter% THEN
  PRINT " File error, no filter defined. Press any key to exit."
  IF GET QUIT                                 : REM we're not doing any more, so close the window
ENDIF
PRINTTAB(0,1) " Noise filter : ";line$        : REM this is the filter title
order% = VAL(FNinput(filter%))                : REM this should be the filter order
PRINTTAB(5,3) " Filter order = ";order%
DIM term(order%/2+1)                          : REM create an array to hold the coefficients
FOR term%=0 TO order%/2
  term(term%) = VAL(FNinput(filter%))         : REM read a coefficient at a time
NEXT
CLOSE# filter%                                : REM done with the filter file

PRINTTAB(0,5) " Horizontal spatial filtering is done on each plane of the file (R, G and B) separately,"
PRINT " vertical filtering is not needed."

REM routine to get the input BMP file name for processing.

in%=0       : REM this is going to be the input file handle
infile$=""  : REM and this will be the file name
out%=0      : REM this is going to be the output file handle
outfile$="" : REM and this will be the file name

DIM of% 75,ff% 255,fn% 255 : REM byte arrays needed for Windows OpenFile routine
!of%=76     : of%!4=@hwnd%  : of%!12=ff% : of%!28=fn%
of%!32=256 : of%!52=6      : REM BB4W stuff for Windows GetOpenFile routine
$fn% = CHR$(0)             : REM this is going to be the file name
$ff% = "YUV 422 image file (*.bmp)" + CHR$0 + "*.bmp" + CHR$0 + CHR$0
SYS "GetOpenFileName", of% TO in%
IF in% THEN
  infile$ = FNnulterm$(fn%)
  outfile$ = LEFT$(infile$, LEN(infile$) - 4) + "-hpfx2.bmp"
  PRINTTAB(0,8) " Input  file = " infile$
  PRINTTAB(0,9) " Output file = " outfile$
ELSE
  PRINTTAB(0,11) " Programme aborted at GetOpen, press any key to exit."
  IF GET QUIT                 : REM we're not doing any more, so close the window
ENDIF

REM Now we can get on with it ...

in% = OPENUP infile$ :REM open YUV bitmap file for reading

IF CHR$(BGET# in%) + CHR$(BGET# in%) <> "BM" THEN
  PRINTTAB(0,14) " This isn't a Windows bitmap file, press any key to exit."
  IF GET QUIT                 :REM we're not doing any more, so close the window
ENDIF

PTR# in% = 10 : start% = FNget4(in%)  : REM size of the header block, where image data starts
PTR# in% = 18 : wide%  = FNget4(in%)  : REM image width in pixels
PTR# in% = 22 : high%  = FNget4(in%)  : REM image height in lines
PTR# in% = 0                          : REM reset ready to start copying the header block

out% = OPENOUT outfile$               : REM open new file for RGB output
FOR p% = 1 TO start%                  : REM copy the header block from input to output file
  BPUT# out%, BGET# in%
NEXT

REM now we can do the actual conversion.

PTR# out% = start%
FOR y% = 1 TO high%                                       : REM beware, images are stored inverted in BMP files
  PTR# out% = FNptr(0, y%, wide%, high%, 3) + start%      : REM each line in turn
  FOR x%= 1 TO wide%
    R = 1/2 : G = 1/2 : B = 1/2                           : REM reset for accummulation, set to mid-grey
    FOR term% = -order%/2 TO order%/2
      p% = x% + term%                                     : REM actual pixel position
      p% = FNmax(p%, 1)                                   : REM avoid going off image to the left
      p% = FNmin(p%, wide%)                               : REM and right
      PTR# in% = FNptr(p%, y%, wide%, high%, 3) + start% : REM read pixels around the target point
      B += (FNdac(BGET# in%) * term(ABS(term%)) * 2)      : REM accumulate values, weighted by the filter
      G += (FNdac(BGET# in%) * term(ABS(term%)) * 2)      : REM and doubled to avoid clipping noise
      R += (FNdac(BGET# in%) * term(ABS(term%)) * 2)
    NEXT
    BPUT# out%, FNadc(B) : BPUT# out%, FNadc(G) : BPUT# out%, FNadc(R)
  NEXT
  PRINTTAB(0,14) " Processing line ";y%
NEXT
CLOSE# in%  : REM finished with input file
CLOSE# out% : REM finished with output file


aspect = wide% / high%                : REM image aspect ratio

scale=1                               : REM scale factor for loaded bitmap file
r% = 1                                : REM flag for "OK"
SYS "GetSystemMetrics", 0 TO wscreen% : REM get the screen width for the actual computer display
SYS "GetSystemMetrics", 1 TO hscreen% : REM and height
IF wide%>wscreen% OR high%>hscreen%-65 THEN
  SYS "MessageBox", @hwnd%, "File too big for the display, scale and load it anyway (colours may be wrong)?", "Load BMP File", 32+1 TO r%
  scale = FNmax(wide% / wscreen%, high% / (hscreen% - 65))
  IF r%=1 THEN
    SYS "SetWindowText", @hwnd%, FNname(outfile$)
  ELSE
    PRINT " Process aborted at file loading stage. Press any key to exit."
    IF GET QUIT                       : REM we're not doing any more, so close the window
  ENDIF
ENDIF

REM set a screen mode to accommodate the inage file, this is Windows stuff

DIM rc% 15                                          : REM data block for screen window size
VDU 23, 22, high% / scale * aspect; high% / scale; 8, 16, 16, 0 : REM don't ask, just don't ask :-)
```

```
SYS "PatBlt", @memhdc%, 0, 0, 1600, 1200, &FF0062
SYS "GetSysColor", 5 TO f%                          : REM look up system colours
COLOUR 15, f%, f%>>8, f%>>16                         : REM define colour 15 in RGB
SYS "GetClientRect", @hwnd%, rc%                    : REM get the display screen size
wwindow% = rc%!8 : hwindow% = rc%!12 + 2            : REM size of window after status bar added
COLOUR 128 + 15 : CLS : REM set white as background colour and clear to it
COLOUR 0                                            : REM black for printing
SYS "GetWindowLong", @hwnd%, -16 TO f%              : REM get window dimensions
SYS "SetWindowLong", @hwnd%, -16, f% OR &40000      : REM don't lock them
SYS "GetClientRect", @hwnd%, rc%                    : REM get window size
VDU 26, 28, 1, hwindow% / 16 - 2, wwindow% / 8 - 2, 1 : REM now set the actual display window for the image
IF scale > 1 SYS "SetStretchBltMode", @memhdc%, 3

REM now we  can load and display the file

OSCLI "Display """ + outfile$ + """ 0,0," + STR$(INT(high% * 2 / scale * aspect)) + "," + STR$(INT(high% * 2 / scale))

END  : REM program ends here, but the window stays open


REM These are standard routines

DEF FNnulterm$(A%) : REM return BB4W string from Windows string (terminated by null)
LOCAL s$
WHILE ?A% <> 0
  s$ += CHR$(?A%) : A% += 1 : REM strip off characters until the first null
ENDWHILE
=s$

DEF FNinput(A%) : REM read a line of text from the file, throw away non-printing characters
LOCAL l$
INPUT# A%, l$
IF ASC(l$) <= 32 : l$ = MID$(l$,2)
IF ASC(RIGHT$(l$, 1)) <=32 : l$ = LEFT$(l$, LEN(l$) -1)
=l$

DEF FNptr(A%,B%,C%,D%,E%) : REM point to pixel at a%,b%, image c%xd%, e% planes
=(D% - B%) * ((E% * C% + E%) DIV 4 * 4) + E% * (A% - 1)

DEF FNget4(A%) : REM get 4 byte number from file
=FNget2(A%) + 256 * 256 * FNget2(A%)

DEF FNget2(A%) : REM get a 2 byte number from file
=(BGET# A%) + 256 * (BGET# A%)

DEF FNmax(A, B) : REM return the greater value
IF A > B : = A
= B

DEF FNmin(A, B) : REM return the greater value
IF A < B : = A
= B

DEF FNadc(A) : REM RGB coder quantiser, return digits
LOCAL A%
A% = 16 + 219 * A
IF A% > 255 : = 255
= A%

DEF FNdac(A) : REM undo coder RGB digitising, return analogue
=(A - 16) / 219
```