```c
//-------------------------4 FIR Sharpening Filter-----------------
#include <math.h>
#include "tiff.h"
#include "allocate.h"
#include "randlib.h"
#include "typeutil.h"

void error(char *name);

int main (int argc, char **argv)
{
  FILE *fp;
  struct TIFF_img input_img, green_img, red_img,blue_img, color_img;
  double **img1,**imgr,**imgb,**img2,**img3,**img4,rho;
  int32_t i,j,pixelg,ii,jj,pixelr,pixelb;

  /* accepts a command line argument specifying the value of rho */
  scanf("%lf", &rho);

  if ( argc != 2 ) error( argv[0] );

  /* open image file */
  if ( ( fp = fopen ( argv[1], "rb" ) ) == NULL ) {
    fprintf ( stderr, "cannot open file %s\n", argv[1] );
    exit ( 1 );
  }

  /* read image */
  if ( read_TIFF ( fp, &input_img ) ) {
    fprintf ( stderr, "error reading file %s\n", argv[1] );
    exit ( 1 );
  }

  /* close image file */
  fclose ( fp );

  /* check the type of image data */
  if ( input_img.TIFF_type != 'c' ) {
    fprintf ( stderr, "error:  image must be 24-bit color\n" );
    exit ( 1 );
  }

  /* Allocate image of double precision floats */
  img1 = (double **)get_img(input_img.width,input_img.height,sizeof(double));
  imgr = (double **)get_img(input_img.width,input_img.height,sizeof(double));
  imgb = (double **)get_img(input_img.width,input_img.height,sizeof(double));
  img2 = (double **)get_img(input_img.width,input_img.height,sizeof(double));
  img3 = (double **)get_img(input_img.width,input_img.height,sizeof(double));
  img4 = (double **)get_img(input_img.width,input_img.height,sizeof(double));

  // /* Initialize the img arrays */
  // for ( i = 0; i < input_img.height; i++ )
  // for ( j = 0; j < input_img.width; j++ ) {
  //   img1[i][j] = 0;
  //   img2[i][j] = 0;
  // }


  /* copy green, red & blue component to double array */
  for ( i = 0; i < input_img.height; i++ )
  for ( j = 0; j < input_img.width; j++ ) {
    img1[i][j] = input_img.color[1][i][j];
    imgr[i][j] = input_img.color[0][i][j];
    imgb[i][j] = input_img.color[2][i][j];
  }


  /* Filter image with the F FIR Sharpening Filter */
  for ( i = 2; i < input_img.height-2; i++ )
  for ( j = 2; j < input_img.width-2; j++ ) {
    // img2[i][j] = (img1[i][j-1] + img1[i][j] + img1[i][j+1])/3.0;
    for ( ii = -2; ii <= 2; ii++ )
    for ( jj = -2; jj <= 2; jj++ ) {
      if (ii == 0 && jj == 0) {
        img2[i][j] = img2[i][j] + (1+rho)*img1[i+ii][j+jj];
        img3[i][j] = img3[i][j] + (1+rho)*imgr[i+ii][j+jj];
        img4[i][j] = img4[i][j] + (1+rho)*imgb[i+ii][j+jj];
      }
      img2[i][j] = img2[i][j] - (rho*1/25)*img1[i+ii][j+jj];
      img3[i][j] = img3[i][j] - (rho*1/25)*imgr[i+ii][j+jj];
      img4[i][j] = img4[i][j] - (rho*1/25)*imgb[i+ii][j+jj];
    }
  }

  /* Fill in boundary pixels */

  // for ( i = 0; i < input_img.height; i++ ) {
  //   img2[i][0] = 0;
  //   img2[i][input_img.width-1] = 0;
  // }

  for ( i = 0; i < 2; i++ )
  for ( j = 0; j < 2; j++ ) {
    for ( ii = -1*i; ii <= 2; ii++ )
    for ( jj = -1*i; jj <= 2; jj++ ) {
      if (ii == 0 && jj == 0) {
        img2[i][j] = img2[i][j] + (1+rho)*img1[i+ii][j+jj];
        img3[i][j] = img3[i][j] + (1+rho)*imgr[i+ii][j+jj];
        img4[i][j] = img4[i][j] + (1+rho)*imgb[i+ii][j+jj];
      }
      img2[i][j] = img2[i][j] - (rho*1/25)*img1[i+ii][j+jj];
      img3[i][j] = img3[i][j] - (rho*1/25)*imgr[i+ii][j+jj];
      img4[i][j] = img4[i][j] - (rho*1/25)*imgb[i+ii][j+jj];
    }
  }

  for ( i = input_img.height-2; i < input_img.height; i++ )
  for ( j = input_img.width-2; j < input_img.width; j++ ) {
    for ( ii = -2; ii < input_img.height-i; ii++ )
    for ( jj = -2; jj < input_img.width-j; jj++ ) {
      if (ii == 0 && jj == 0) {
        img2[i][j] = img2[i][j] + (1+rho)*img1[i+ii][j+jj];
        img3[i][j] = img3[i][j] + (1+rho)*imgr[i+ii][j+jj];
        img4[i][j] = img4[i][j] + (1+rho)*imgb[i+ii][j+jj];
      }
      img2[i][j] = img2[i][j] - (rho*1/25)*img1[i+ii][j+jj];
      img3[i][j] = img3[i][j] - (rho*1/25)*imgr[i+ii][j+jj];
      img4[i][j] = img4[i][j] - (rho*1/25)*imgb[i+ii][j+jj];
    }
  }

  for ( i = 0; i < 2; i++ )
  for ( j = input_img.width-2; j < input_img.width; j++ ) {
    for ( ii = -1*i; ii <= 2; ii++ )
    for ( jj = -2; jj < input_img.width-j; jj++ ) {
      if (ii == 0 && jj == 0) {
        img2[i][j] = img2[i][j] + (1+rho)*img1[i+ii][j+jj];
        img3[i][j] = img3[i][j] + (1+rho)*imgr[i+ii][j+jj];
        img4[i][j] = img4[i][j] + (1+rho)*imgb[i+ii][j+jj];
      }
      img2[i][j] = img2[i][j] - (rho*1/25)*img1[i+ii][j+jj];
      img3[i][j] = img3[i][j] - (rho*1/25)*imgr[i+ii][j+jj];
      img4[i][j] = img4[i][j] - (rho*1/25)*imgb[i+ii][j+jj];
    }
  }

  for ( i = input_img.height-2; i < input_img.height; i++ )
  for ( j = 0; j < 2; j++ ) {
    for ( ii = -2; ii < input_img.height-i; ii++ )
    for ( jj = -1*i; jj <= 2; jj++ ) {
      if (ii == 0 && jj == 0) {
        img2[i][j] = img2[i][j] + (1+rho)*img1[i+ii][j+jj];
        img3[i][j] = img3[i][j] + (1+rho)*imgr[i+ii][j+jj];
        img4[i][j] = img4[i][j] + (1+rho)*imgb[i+ii][j+jj];
      }
      img2[i][j] = img2[i][j] - (rho*1/25)*img1[i+ii][j+jj];
      img3[i][j] = img3[i][j] - (rho*1/25)*imgr[i+ii][j+jj];
      img4[i][j] = img4[i][j] - (rho*1/25)*imgb[i+ii][j+jj];
    }
  }


  // /* Set seed for random noise generator */
  // srandom2(1);

  // /* Add noise to image */
  // for ( i = 0; i < input_img.height; i++ )
  // for ( j = 1; j < input_img.width-1; j++ ) {
  //   img2[i][j] += 32*normal();
  // }

  /* set up structure for output achromatic image */
  /* to allocate a full color image use type 'c' */
  get_TIFF ( &green_img, input_img.height, input_img.width, 'g' );
  get_TIFF ( &red_img, input_img.height, input_img.width, 'g' );
  get_TIFF ( &blue_img, input_img.height, input_img.width, 'g' );

  /* set up structure for output color image */
  /* Note that the type is 'c' rather than 'g' */
  get_TIFF ( &color_img, input_img.height, input_img.width, 'c' );

  /* copy green, red & blue component to new images */
  for ( i = 0; i < input_img.height; i++ )
  for ( j = 0; j < input_img.width; j++ ) {
    pixelg = (int32_t)img2[i][j];
    pixelr = (int32_t)img3[i][j];
    pixelb = (int32_t)img4[i][j];

    if(pixelg>255) {
      green_img.mono[i][j] = 255;
```

```c
    }
    else {
      if(pixelg<0) green_img.mono[i][j] = 0;
      else green_img.mono[i][j] = pixelg;
    }

    if(pixelr>255) {
      red_img.mono[i][j] = 255;
    }
    else {
      if(pixelr<0) red_img.mono[i][j] = 0;
      else red_img.mono[i][j] = pixelr;
    }

    if(pixelb>255) {
      blue_img.mono[i][j] = 255;
    }
    else {
      if(pixelb<0) blue_img.mono[i][j] = 0;
      else blue_img.mono[i][j] = pixelb;
    }
  }

  // /* Illustration: constructing a sample color image -- interchanging the red and green components from the input color image */
  // for ( i = 0; i < input_img.height; i++ )
  //     for ( j = 0; j < input_img.width; j++ ) {
  //         color_img.color[0][i][j] = input_img.color[1][i][j];
  //         color_img.color[1][i][j] = input_img.color[0][i][j];
  //         color_img.color[2][i][j] = input_img.color[2][i][j];
  //     }

  /* Illustration: constructing a sample color image -- put 3 image (green,red,blue) into 1 image */
  for ( i = 0; i < input_img.height; i++ )
      for ( j = 0; j < input_img.width; j++ ) {
          color_img.color[0][i][j] = red_img.mono[i][j];
          color_img.color[1][i][j] = green_img.mono[i][j];
          color_img.color[2][i][j] = blue_img.mono[i][j];
      }

  // /* open green image file */
  // if ( ( fp = fopen ( "green.tif", "wb" ) ) == NULL ) {
  //   fprintf ( stderr, "cannot open file green.tif\n");
  //   exit ( 1 );
  // }

  // /* write green image */
  // if ( write_TIFF ( fp, &green_img ) ) {
  //   fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
  //   exit ( 1 );
  // }

  // /* close green image file */
  // fclose ( fp );


  /* open color image file */
  if ( ( fp = fopen ( "color.tif", "wb" ) ) == NULL ) {
      fprintf ( stderr, "cannot open file color.tif\n");
      exit ( 1 );
  }

  /* write color image */
  if ( write_TIFF ( fp, &color_img ) ) {
      fprintf ( stderr, "error writing TIFF file %s\n", argv[2] );
      exit ( 1 );
  }

  /* close color image file */
  fclose ( fp );

  /* de-allocate space which was used for the images */
  free_TIFF ( &(input_img) );
  free_TIFF ( &(green_img) );
  free_TIFF ( &(red_img) );
  free_TIFF ( &(blue_img) );
  free_TIFF ( &(color_img) );

  free_img( (void**)img1 );
  free_img( (void**)img2 );
  free_img( (void**)img3 );
  free_img( (void**)img4 );
  free_img( (void**)imgr );
  free_img( (void**)imgb );

  return(0);
}

void error(char *name)
{
    printf("usage:  %s  image.tiff \n\n",name);
    printf("this program reads in a 24-bit color TIFF image.\n");
    printf("It then horizontally filters the green component, adds noise,\n");
    printf("and writes out the result as an 8-bit image\n");
    printf("with the name 'green.tiff'.\n");
    printf("It also generates an 8-bit color image,\n");
    printf("that swaps red and green components from the input image");
    exit(1);
}


```