Lab 5: Image file creation

Goals

Construct a C language program that will produce a images of the flags of France, Germany, and Lithuania.

Background

Image files

- Images (e.g. digital photos) consist of a rectangular array of *discrete picture elements* called *pixels.*

- An image consisting of 200 rows of 300 pixels per row contains 300 x 200 = 60,000 individual pixels. The *height* of the image in pixels is the number of pixel rows and the *width* is the number of pixels in each row. A color image requires 3 bytes per pixel or 180,000 total bytes!

- The Portable PixMap *(.ppm)* format is a particularly simple way used to encode a rectangular image (picture) as uncompressed data file.

- The *.ppm* file can viewed with a number of tools including *xv* and *gimp.* M$ Window$ systems have no built-in .ppm viewer. Several freeware viewers are available on the web though. See (www.irfanview.com) for example.

- Other well known formats include JPEG (.jpg), TIFF (.tif), GIF (.gif), bitmap (.bmp), and PNG (.png)

PPM file format

- **ppm files:** *ppm* is a very simple image file format. A *ppm* image consists of two components:

  1. **The header:** the header contains
     1) a *label* to identify the file format as a color ppm file ("P6"),
     2) the *width* of the image in pixels,
     3) the *height* of the image in pixels and
     4) the maximum pixel value (*always 255*).
     5) the header ends with a *single \n* newline character.

  2. **Binary image data:** the data consists of unsigned char (eight-bit == one-byte) binary values defining the color of each pixel.

- **Example ppm header**: The header of a color image of 800 pixels wide and 600 pixels high has the following format

```
P6
800 600 255
```

- **Pixel data format:**

  - each pixel consists of 3 one-byte values of type *unsigned char.*
  - the first three bytes of image data in the .ppm file define the color of the pixel in the upper left corner of the image
  - the last three bytes in the file define the color of the pixel in the lower right
  - pixel values defining each horizontal row of the image are adjacent
  - *NO spaces, tabs, newlines may be embedded in the file.*

- **red/green/blue image encoding**
  - this format is called RGB. The three bytes of each pixel represent the color intensities of the:
    - red component
    - green component
    - blue component
  - (255, 0, 0) is bright red
  - (0, 255, 0) is bright green
  - (0, 0, 255) is bright blue

- **Colors are additive**

  - (255, 255, 0) = red + green = bright yellow
  - (255, 0, 255) = red + blue = magenta (purple)
  - (0, 255, 255) = blue + green = cyan (turquoise)
  - (255, 255, 255) = red + green + blue = white
  - when *red* == *green* == *blue* a gray “color” is produced

- **Writing a pixel value**

  - The *%c* format code tells fprintf() *NOT to convert* the value being printed to ASCII format.
  - Since pixel values are binary, the *%c code must be used*.
  - The following statement can be used to write a red pixel

```
fprintf(stdout, "%c%c%c", 255, 0, 0);
```

  - NO spaces or newlines are permitted in the format string!!

Assignment:

Write a program called lab5.c that is capable of creating an image of the French, German, or Lithuanian flag. Two integer values should be read from the standard input:

country_code:	0 - France 1 - Germany 2 - Lithuania
width:	The width of the flag in pixels.

You should use the proper colors as defined in Wikipedia based upon a google search for "Flag of France" etc. You should use the *proper ratio of height to width* as specified in Wikipedia to compute the height of the image you produce.

Use the *make_pixel()* function provided in the class notes to write individual pixels:

```
void make_pixel(
int r,            // red intensity
int g,            // green intensity
int b)            // blue intensity
{
   fprintf(stdout, "%c%c%c", r, g, b);
}
```

Use the *make_ppm_header()* function provided in the class notes to write individual pixels:

```
void make_ppm_header(
int width,
int height)
{
   fprintf(stdout, "P6\n");
   fprintf(stdout, "%d %d %d\n",
                    width, height, 255);
}
```

Use the following *main()* function.

```
#include <stdio.h>

int main()
{
   int width;
   int country_code;

/* Read image dimensions and pixel color */

   fscanf(stdin, "%d %d", &country_code, &width);
   fprintf(stderr, "Making country %d width %d \n",
            country_code, width);

/* Write the image data */

    make_ppm_image(country_code, width);

   return(0);
}
```

For full credit:

Use functions in such a way that your program does *not* contain:

(1) if nested within while loop
(2) while loop nested within while loop
(3) while loop nested within if
(4) for loop

Your program should compile WITHOUT warnings with gcc -Wall -g lab5.c

Turn In Work

Show your TA that you completed the assignment. Then turn in your lab4.c program using the command:

sendlab.101.*section_number* 5 lab5.c