DR ANTON GERDELAN

WARM-UP STARTER

WARM-UP ASSIGNMENT

- Write an image to a file without using a library, and draw a diagonal line over it
- C-style memory allocation
- Input?
- Output?
- Good use of loops and arrays
- Some sort of visual feedback!
- Draw something creative

HOW TO START

- Look up .ppm image format specification
  - use the ASCII format, not the binary version
- What are the contents?
  - header (some required text about format, dims. etc.)
  - body (with formatting)
- Do we know how to write to an ASCII file in C?

OUTPUT FILE

- Try directly writing the example file or blank file first
  - don't worry about dynamic code yet
  - does it open in an image editor?
  - do we get the colours that we expected?

SHALL I CODE THIS LIVE OR WRITE ON THE WHITEBOARD?

OUTPUT FILE

- Decide on image size
    - width and height in pixels
    - pixels are usually combo red, green, blue channels
        - antongerdelan.net/colour
    - colour depth per channel - 1 or 2 bytes each? (use 1)
        - i.e. value for red: 0-255, green: 0-255, blue: 0-255

OUTPUT FILE

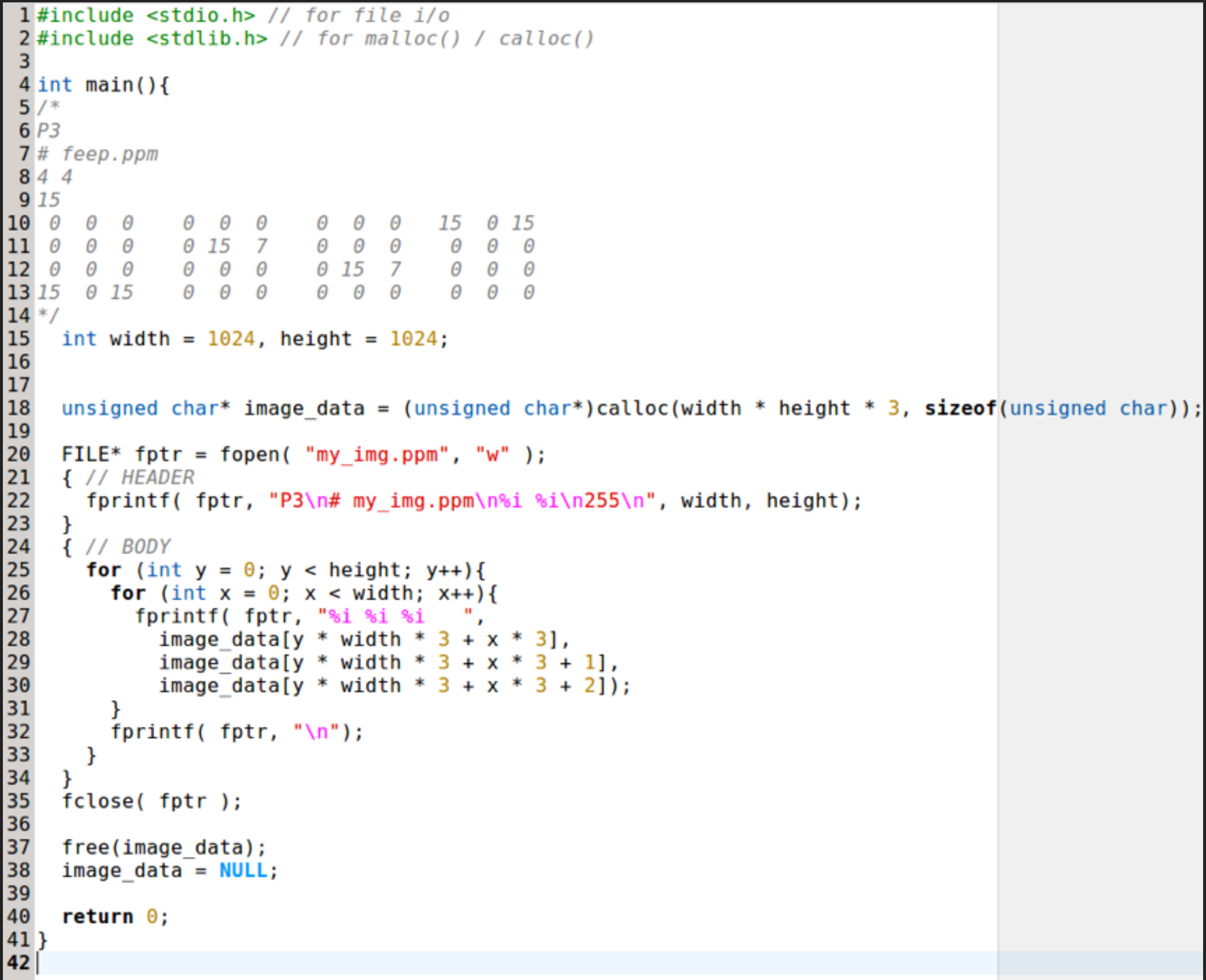
- How much data is in each pixel?
- We could write each pixel directly to file e.g. based on a mathematical function
- We could also allocate memory for all the pixels and modify this as we like before writing to file (do this)
- How much memory do we need to represent the image?

MEMORY FOR THE IMAGE

- How do we allocate space for the image in memory?
- Is memory initialised to zero?
    - `calloc()`
    - if all memory is 0 0 0 0… what colour?
- What data structure is our memory?
- How do we determine which bytes image[row 10][column 3] correspond to?

WRITING THE FILE FROM MEMORY

- assuming we have our image's block of memory
- how do we write it to a file
  - consider format of PPM (output)
  - and data layout of image (input)
- `fprintf()`
- loops?

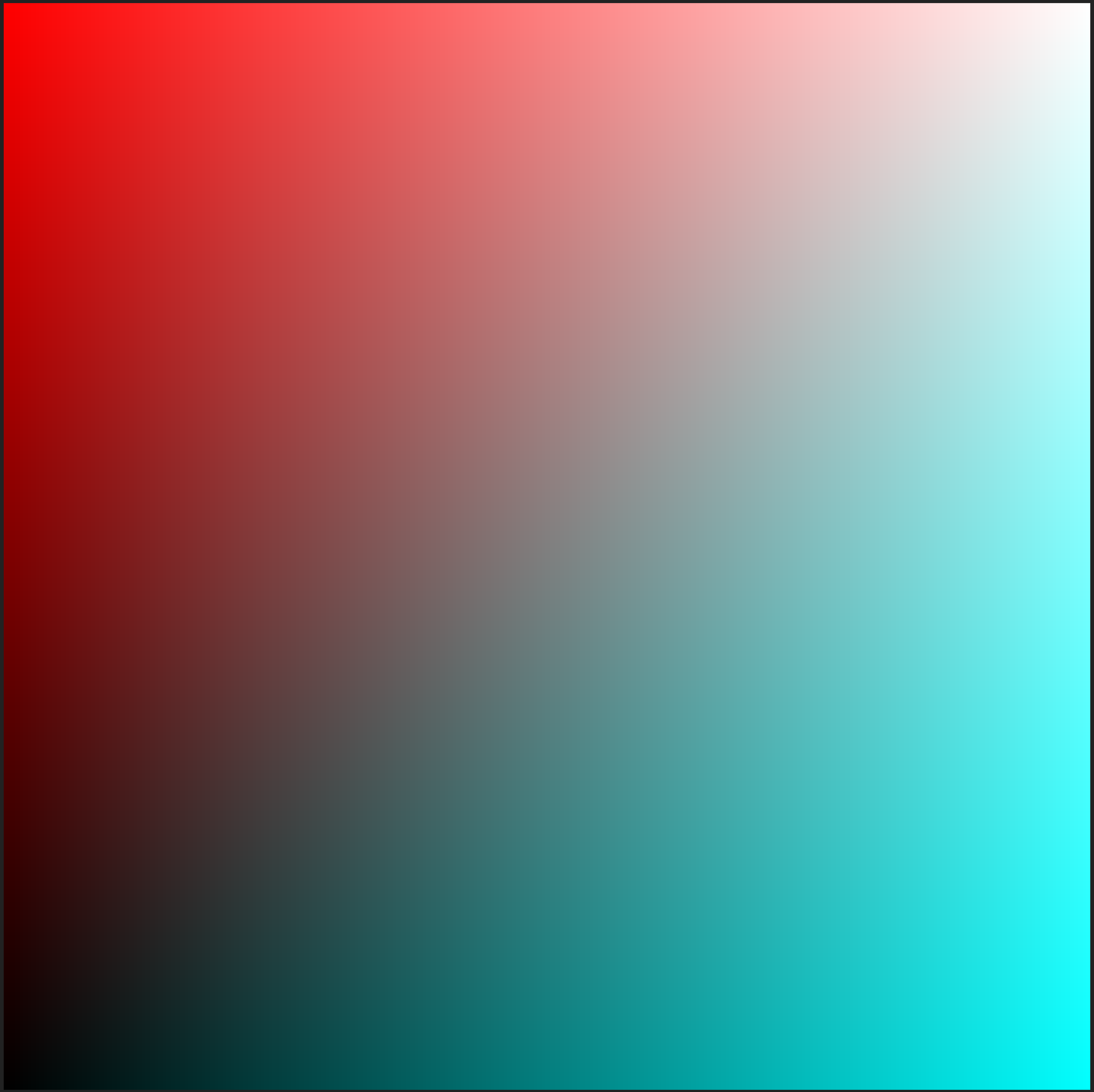


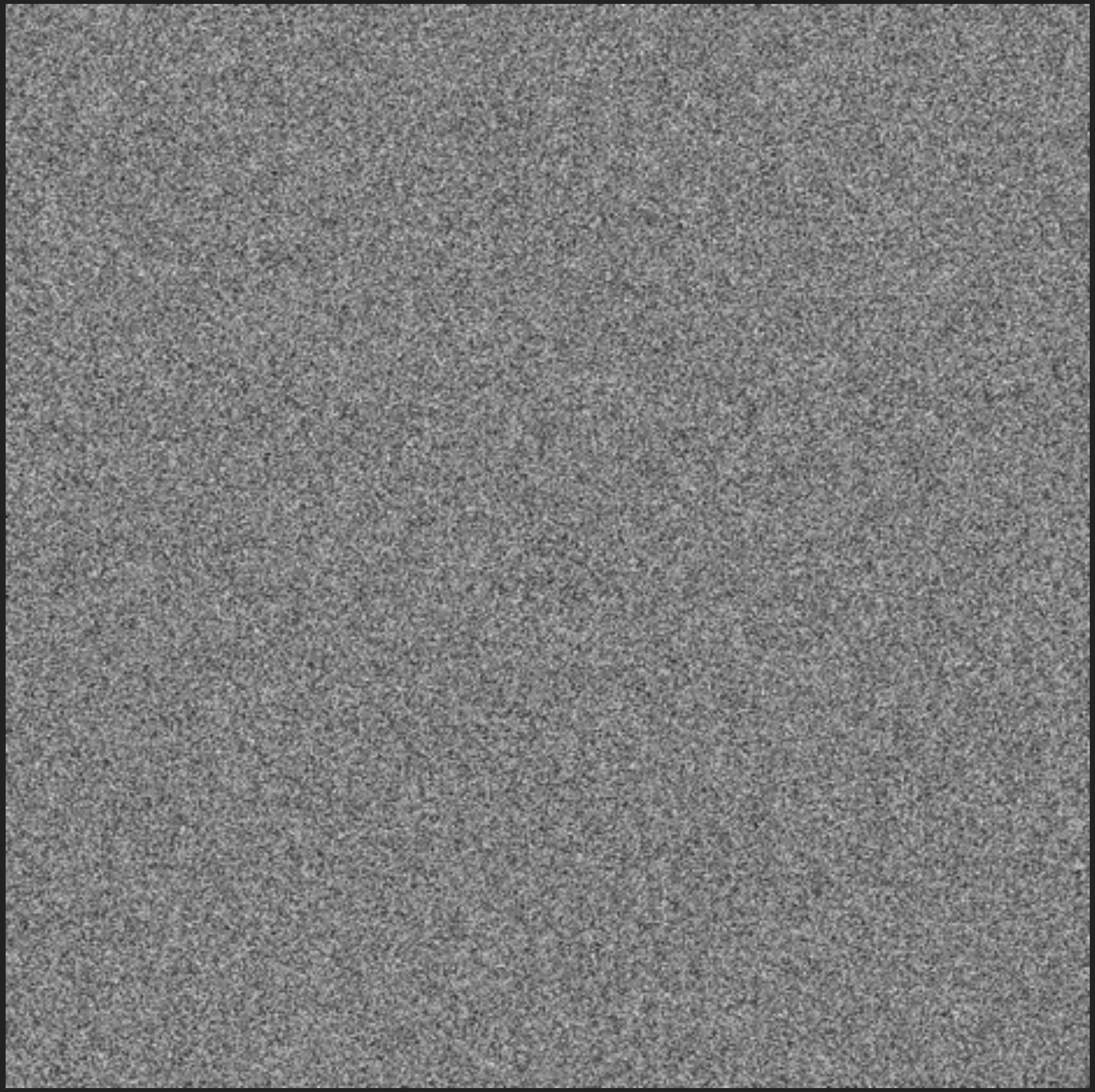
```c
#include <stdio.h> // for file i/o
#include <stdlib.h> // for malloc() / calloc()

int main(){
/*
P3
# feep.ppm
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
*/
  int width = 1024, height = 1024;


  unsigned char* image_data = (unsigned char*)calloc(width * height * 3, sizeof(unsigned char));

  FILE* fptr = fopen( "my_img.ppm", "w" );
  { // HEADER
    fprintf( fptr, "P3\n# my_img.ppm\n%i %i\n255\n", width, height);
  }
  { // BODY
    for (int y = 0; y < height; y++){
      for (int x = 0; x < width; x++){
        fprintf( fptr, "%i %i %i   ",
          image_data[y * width * 3 + x * 3],
          image_data[y * width * 3 + x * 3 + 1],
          image_data[y * width * 3 + x * 3 + 2]);
      }
      fprintf( fptr, "\n");
    }
  }
  fclose( fptr );

  free(image_data);
  image_data = NULL;

  return 0;
}

```

maybe t

TRICKY THINGS

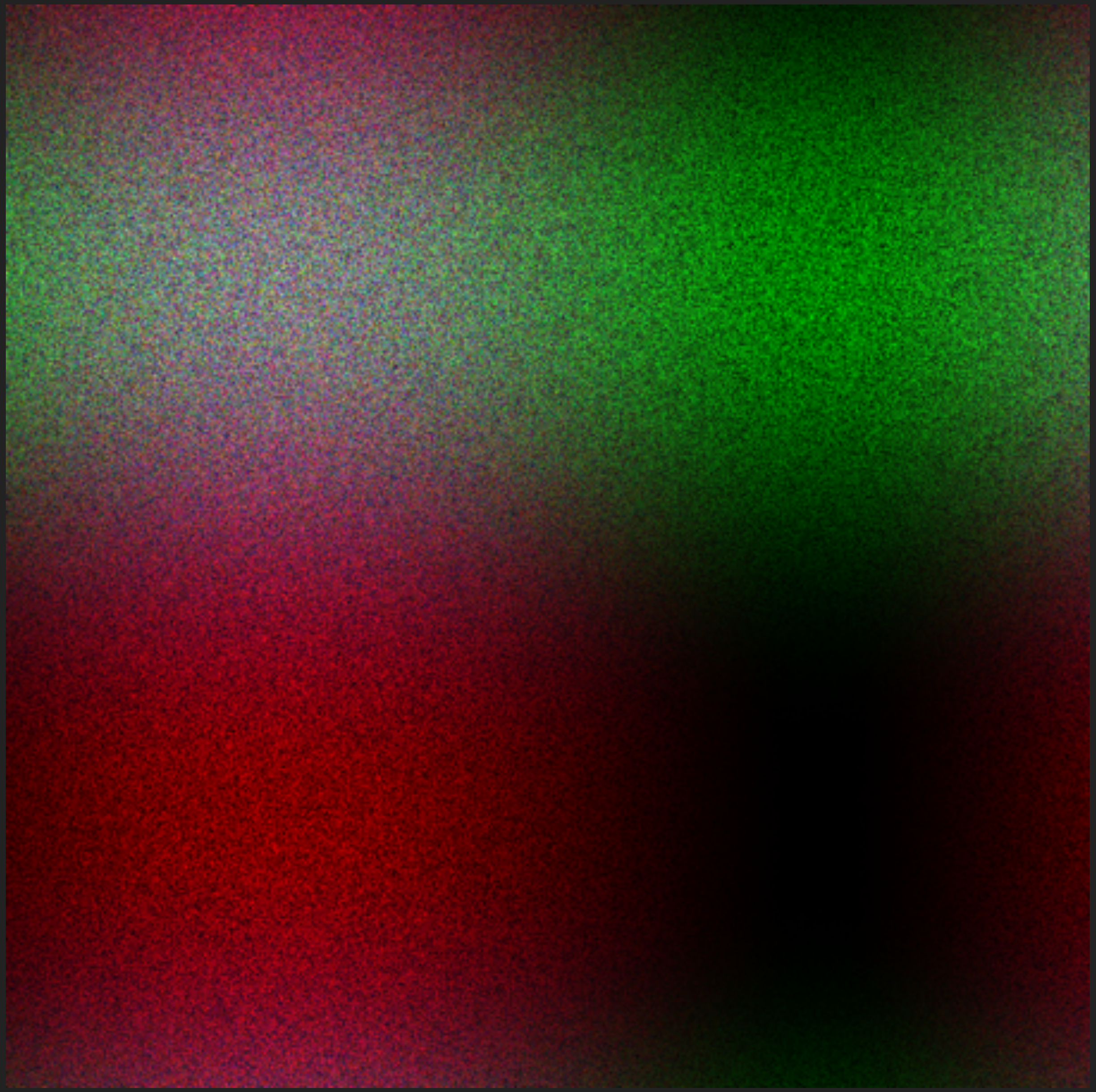
- header format things - use a different format if you like
- determining the index of
  - red, green, blue values to write
- remember to leave a space after each value
- where to print newline after rows
- so, where, and how can we modify the image data before writing it to the file?





noise using the `rand()` function

and some `sin()` waves using x and y as input



DRAWING A DIAGONAL LINE

- a single pixel-thick line over the whole image is fine
- there are some interesting algorithms for drawing lines between 2 given points
  - Bresenham (1962) - efficiency
  - Xiaolin Wu (1991) - efficiency and anti-aliasing
- easy to make mistakes by forgetting that there are 3 channels per pixel - and you get visual feedback

SKILLS YOU CAN CHECK OFF ON COMPLETION

- basic C programming
- dynamic memory allocation and freeing
- pointers
- loops, arrays
- reasoning about data size
- know writing images and file formats isn't that hard
- didn't need to use anyone else's frameworks/libraries

HOW – A PAINT PROGRAM?

- same concepts
- need to use operating system's windowing library
  - windows.h, Cocoa, X
  - SDL2 library etc.
- or use the web (canvas2D etc)
- output is now an image or canvas used by the display area
- input may be mouse coordinates in x,y

IMAGE CONVERTER / PHOTO FILTER

- support other image file formats
  - RAW, .tif
  - libpng
  - stb_image (really cool little lib)
- load from file -> same block of memory -> other file format
- could you flip an image upside down? colour filter?
- add a cut-out image over someone's Instagram photo?

HOW TO DEAL WITH PROBLEMS

- don't panic!
- isolate the problem
- find the problem
  - expectations do not match results?
  - hand calculate expectations
  - use a **debugger** to step through code (or get a **backtrace** after a **segfault**)
  - find the discrepancy
- re-read instructions/man pages/examples
- ask for help

THINGS TO THINK ABOUT

- we have a 2d image but
- our data structure was 1d array
- was it good enough; easy to work with, fast, size?
- how big are the output files compared to e.g. PNG?
- if we have an inner and outer loop, does it matter which one is y and which one is x?
- how do binary file formats work?

COMING SOON

- tutorial - some sample problems and C refresher stuff
- lab - start the assignment, ask for help
- next week
  - data structures: linked-lists, stacks, queues
  - introduction to fundamental algorithms
  - finish warm-up assignment