# Assignment 0 - Warm-Up Lab

Due: 7 Oct 2016

Grade: not graded but feedback given.

Dr Anton Gerdelan gerdela@scss.tcd.ie, Trinity College Dublin

## Purpose of Assignment and Lab

- refresh C programming basics
    - basic compiler skills
    - loops and arrays
    - dynamic memory allocation
    - pointers and addresses
    - functions
- ensure that you can log in/submit work etc.
- create a useful data visualisation tool

## Deliverables

- A pdf containing
    - your final output image(s)
    - a brief summary of your solution
    - your opinion on a few pros/cons of the PPM format
- Your .c file (with your name and date in a comment in the top).
- Do not submit IDE project/solution files, or the compiled binary.

Please do not compress these into a .zip - it's harder for the demonstrators to open in the web system.

## Instructions

You are to write a simple C program that allocates some memory, manipulate that memory using built-in functions and your own loops, and then visualise that memory by writing it out to an image file. We will not be using C++ memory allocation (the `new` keyword) for this assignment - we want to dig deeper. We won't be using any third-party libraries.

See Lecture 3 for example code if you need a starter.

### Task 1

- Set up an environment to compile a simple C or C++ "Hello World" type program. You may use any compiler/IDE software.

The labs should have at least gcc (GNU C Compiler) on the command line and Visual Studio. You should know how to compile in debug mode and use a debugger to step through code. If you're using GCC then Visual Studio Code is a nice, light-weight visual debugger. We can do a tutorial for this if it's a popular request.

## Task 2

- Dynamically allocate a block of memory large enough to store an image with dimensions of your choosing.
- Write a function that will write the image memory to a file in PPM format.
- Make sure that you can open the image file in an image viewer.
- Modify your program - initialise your memory to represent a blank white image.

Use either the `malloc()` or `calloc()` function to allocate an array of bytes to store your image. You may define a width and height variable for your image size. Each pixel in your image requires 3 bytes - one for each of the red, green, and blue channels, respectively. The `unsigned char` data type is a good representation of 1 byte in C. Use the `memset()` function to clear the entire image to white. Use `free()` to deallocate the memory before the program exits (but after writing to a file!). Note that this memory will be automatically freed anyway on progam exit, but it's a good habit to free everything that we allocate as it will help us spot genuine memory leaks later.

PPM is a very simple image format that can viewed by most image-viewing software, and can be written as an ASCII text file.

The specification for the format may be found at http://netpbm.sourceforge.net/doc/ppm.html You may use the binary version of PPM (P6) if you want but I suggest that the ASCII version is better to start with (P3).

Your function should have a declaration similar to:

```
void write_ppm(const unsigned char* image_data, int w, int h);
```

Where w and h are a width and height, respectively, in pixels. image_data is an array of bytes that you will create in memory, containing the colours of each pixel. You can use C's FILE i/o or C++ streams to write your image data to the file, but you will need to write a few lines of header information at the top of the file first.

If your function occupies more than 20 lines of code think about simplifying it.

## Task 3

Modify your program:

- Write a loop or loops to draw a diagonal line from one corner of the image to another.
- Come up with a programmatic way to draw something more creative into the image.

You may need to tell your image viewing software to disable anti-aliasing/blurring to see the pixels clearly.

# Comments, Suggestions

- If this is easy, good. If not, ask for help - you may need to schedule some practice time!
- If your program crashes with a segfault it almost certainly means that you are accessing memory outside the bounds of what you have allocated. Did you remember that each pixel needs 3 bytes of memory?
- Compilers usually look at file extensions to determine if they should compile in C or C++ mode. i.e. main.c, rather than main.cpp.
- If your program is over 50 lines of code, think about simplifying it.

- Have a good grasp of C memory functions; `malloc()`, `calloc()`, `free()`, `memset()`, `memcpy()` etc. and other basic C functions; `assert()` file I/O etc.
- You may have to look up these functions to find out which header file to use; e.g. `malloc()` is in `stdlib.h` but `memcpy()` is in `string.h`.
- On Unix-derived operating systems you can type `man memcpy` etc. To get a manual page for that function. There are good websites too; cprogramming.com, cplusplus.com. MSDN is also pretty good for Windows implementations of the standard C and C++ libraries.
- I usually prefer C to C++ and set my compiler to build in C99 mode, which is friendlier to work with than C89, which may be the default still.
- Visual Studio doesn't have excellent C99 support last time I checked, so it might be easier to just use C-like C++.
- For a nice library to write to other formats of image look for stb*image*write
- If you write images out in a sequence with numbered filenames you can turn them into a .gif or movie file.
- Consider how you might write a function to plot a sine wave onto the image space.
- Being primarily visual beings, a lot of computing and data can be easier to understand or demonstrate the output of if you can write an image, chart, or animation. You may also find it convenient to represent (or paint) input instances in an image.