exercise 0

these are my solutions to the first (optional) exercise set of TDT4200.

i recommend using a PDF-reader like zathura, for no particular reason other than that it is cool.

this document was created using typst.

how to run my code

the project utilizes nix to manage dependencies.

with it (and with flakes enabled), you can simply run `nix develop` to enter a development shell with the necessary dependencies, as specified in `flake.nix` and `flake.lock`. in this shell you can build the bmp-image using `make run` or to see it immediately, using feh, you can run `make show`.

theory

1) what is the purpose of a Makefile?

a Makefile is like a shell-script that performs some pre-defined tasks. it is particularly useful as a **build tool** for complex compilation pipelines in development environments relying on such compiled languages like C.

`make show` is an example of a neat short-hand to compile and run my program, but also finally displaying the image result to me using the lightweight CLI feh.

2) what is a pointer in C?

a **pointer** in C is a memory address equipped with a primitive type, determining the semantics of **pointer arithmetic**, i.e. when you increment the pointer. for example, if you increment a pointer with a type of size 8 bytes, it will jump to the memory address 8 bytes forward. a **void pointer** is a true wildcard pointer, having 'no type', thus being able to point to any memory, disregarding alignment.

3) what does the compile flag `-O3` do?

GCC, like most compilers, have lots of flags that alter their behavior. given the age of C and the amount of clever people in the world that have dedicated time to crafting a perfect compiler, GCC is capable of analyzing the code to perform optimizations. `gcc -O3 source.c` means to compile a source file with **aggressive optimizations**. examples of this could be unrolling loops, recognizing patterns or using more specialized (and faster) instructions (if available).

note that there have been examples of compiler optimizations introducing weird bugs (i think), due to slight alterings of internal semantics.

4) how do you pass a value by reference?

passing a value by reference is the preferred way of handling data transfer in C, as opposed to other languages, where data may be passed to a function as a **pure copy**. copying memory is expensive, both time-wise and space-wise. it scales linearly, so thus if you have millions of data points, and you wish to pass it around lots, copying it each time could be considered unoptimal. that's why you wish to refer to your data using pointers, also known as **references**.

じゃあね、またね。