# Software Technology of Internet of Things

## C Exercises: Into the Muck

Aslak Johansen <asjo@mmmi.sdu.dk>

Feb 20, 2023

## Quest 1: Making that First Compile (5XP)

> ‘When the first caveman programmer chiseled the first program on the walls of the first cave computer, it was a program to paint the string ‘Hello, world’ in Antelope pictures. Roman programming textbooks began with the ‘Salut, Mundi’ program. I don’t know what happens to people who break with this tradition, but I think it’s safer not to find out.’
>
> The Linux Kernel Module Programming Guide

Today, you start out by following the footsteps of the many. Your quest is to write that very first program by making the following code appear in a file:

```c
#include <stdio.h>

int main (int argc, char* argv[])
{
  printf("Hello, World\n");
  return 0;
}
```

You must prove that you can compile it, execute it, and observe its output.

## Quest 2: Offsetting the Index (10XP)

Your next challenge is to fool the minds of the feeble by commanding a 0-indexed array as if it was a 1-indexed one. This technique will allow us to save a bit of space would we never need to index using zero.

1. Create a new file `index0.c`.
2. Use a `define` precompiler directive to define a constant named `SIZE` to have a value of 10.
3. Add a main function that returns 0.
4. In the main function you declare a variable `a` to be an `int[]` and initialize it by allocating space for `SIZE` elements.
5. Then you use a loop to iterate through all indices and for each one use a `printf` to print out the index as well as the data on that position of the array.
6. Explain the printouts you see.
7. Copy the file to `index0b.c`, and continue the work on this file.
8. Modify to code to make sure that position $i$ has the value $i$ for all `SIZE` positions.
9. Verify by executing the code and observing the outputs.
10. Copy the file to `index1.c`, and continue the work on this file.
11. Add a variable named `b` of type `int*` that refers to the address of element 0 of `a`.
12. Decrement `b` by 1 before the loop from step 5.
13. Adjust the loop from step 5 to use `b` instead of `a`.
14. Verify that `b` now acts a 1-indexed version of `a`.
15. Add a final line to the main function printing out the value of `b[0]`.
16. Explain the results.

# Quest 3: In Support of a Hero (25XP)

Samantha Groves, first of her name, has trained hard to become a highly skilled social engineer. Ms. Groves is about to set out to do heroics in a large number of Danish cities, but she realizes that time spent on the road is time not spent doing heroics. You must help her find the shortest path through the full list of cities and back again! The cities, and distances between them, are expressed in the following data structures:

```
#define CITY_COUNT (15)

char* citynames[CITY_COUNT] = {
    "Esbjerg",
    "Helsingør",
    "Herning",
    "Horsens",
    "Kolding",
    "København",
    "Næstved",
    "Odense",
    "Randers",
    "Roskilde",
    "Silkeborg",
    "Vejle",
    "Viborg",
    "Aalborg",
    "Århus",
};

// source: https://dk.afstand.org
int distances[CITY_COUNT][CITY_COUNT] = {
    {  0, 269,  82,  98,  65, 260, 211, 123, 149, 230, 105,  74, 126, 198, 134},
    {269,   0, 226, 173, 206,  40, 105, 157, 166,  55, 191, 196, 207, 200, 150},
    { 82, 226,   0,  63,  79, 230, 202, 121,  76, 202,  36,  59,  45, 117,  77},
    { 98, 173,  63,   0,  48, 172, 139,  62,  68, 142,  40,  26,  75, 132,  40},
    { 65, 206,  79,  48,   0, 196, 148,  59, 114, 165,  77,  25, 110, 176,  88},
    {260,  40, 230, 172, 196,   0,  71, 141, 180,  31, 196, 190, 218, 224, 156},
    {211, 105, 202, 139, 148,  71,   0,  89, 174,  50, 175, 150, 204, 232, 142},
    {123, 157, 121,  62,  59, 141,  89,   0, 121, 110, 102,  64, 136, 186,  86},
    {149, 166,  76,  68, 114, 180, 174, 121,   0, 156,  44,  90,  42,  65,  36},
    {230,  55, 202, 142, 165,  31,  50, 110, 156,   0, 169, 160, 193, 206, 130},
    {105, 191,  36,  40,  77, 196, 175, 102,  44, 169,   0,  53,  35,  99,  41},
    { 74, 196,  59,  26,  25, 190, 150,  64,  90, 160,  53,   0,  86, 151,  65},
    {126, 207,  45,  75, 110, 218, 204, 136,  42, 193,  35,  86,   0,  72,  63},
    {198, 200, 117, 132, 176, 224, 232, 186,  65, 206,  99, 151,  72,   0, 101},
    {134, 150,  77,  40,  88, 156, 142,  86,  36, 130,  41,  65,  63, 101,   0},
};
```

# Quest 4: The Final Test (10XP)

In the final test you must analyze the following piece of code in order to understand what its function is, and how it works:

```c
#include <stdio.h>
#include <stdlib.h>

#define LENGTH (12)

int main (int argc, char* argv[]) {
  int* values = (int*) malloc(LENGTH*sizeof(int));

  for (int i=0 ; i<LENGTH ; i++) {
    values[i] = i;
  }

  for (int i=0 ; i<LENGTH/2 ; i++) {
    printf("%u: %d\n", i, (int)(((long*)values)[i]));
  }

  return 0;
}
```