Exercise C2: First program

Software

For this task you need to install GSL, the GNU Scientific Library. You can think of this as the C version of Python's Numpy and Scipy libraries. To do this on the stuDAT computers, run the included installation script as:

```
bash ./install_gsl.sh
```

Note that after installing GSL you must reload your shell. Do this either by typing

```
source ~/.bashrc
```

or simply by closing and opening a new terminal window.

This should work on other Linux computers as well, however, some package managers have it in their repositories. This script does not work on Mac or Windows computers, however!

There is example code how to generate random numbers using GSL in the file gsl_example.c. compile as

```
gcc gsl_example.c -lgsl -lcblas -O0 -o gsl_example
```

Introduction

In this exercise you will have to start coding yourself, the program that you will have to write covers some topics that will help you throughout the course. For example, how to write your data to a file. In particular, it will prepare you for the first homework problems.

Dynamic memory allocation and arrays

- Write a program with a function that computes the scalar product between two arrays. In the main function, read a length from the command line and create two vectors of that length for which you calculate the scalar product. Hint: Allocate the arrays dynamically and use the scanf function to read an int representing the length from the command line.
- Write a function that dynamically allocates a 2D array and use it to create an nx3 array filled with doubles of your choice. Interpreting each row in this array as the coordinate vector of a point in space, write a function that calculates the distance between two given points.
- Write the matrix to a file in a csv format.

Header files and separate sources

Header files

Header files contains function prototypes for its corresponding C file. For example, if you have a C file, linalg.c, that can e.g. calculate the scalar product. Then, if a second C file, main.c, wants to calculate the scalar product it has to include the linalg header file. Remember that header files should have include guards. Include guards makes sure that you are not including the header file twice in a C file.

This could looks something like,

linalg.h:

```
#pragma once // This is the include guard
double scalar_product(double *, double *);
```

linalg.c:

```
double scalar_product(double *a, double *b){
...
}
```

main.c:

```
// "" looks for the linalg.h file in the current directory
// <> looks for the linalg.h file in the system libraries
// Therefore, use "" for user defined header files
#include "linalg.h"

int main()
{
    ...
    scalar_product(a, b);
    ...
}
```

Rewrite your code from the previous exercise so that the two functions (scalar product and pairdistance) are located in a separate C-file. Create a corresponding header file where you put the function prototypes. Recompile your program and make sure that everything works.

Using external libraries, Makefiles

Use GSL to generate a large array filled with random numbers drawn from the uniform distribution on [0,1]. Write the array to a file and import the data into Python and make a histogram to verify that the distribution is indeed uniform. Using the template Makefile located in this repo to simplify linking the GSL libraries.