Programming in C: Header files

Table of Contents

1 Introduction 2

2 Assignment 3

3 Possible questions 4

3.1 Tips 4

3.2 Assignment 4



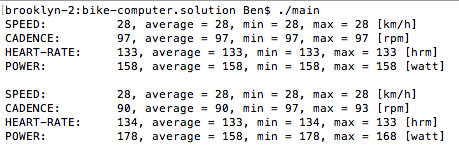
*Figure 1: picture of a bike-computer, s*ource photo: [link](https://cdn.roadcyclinguk.com/featured_image/5ab94697beb4f.jpg)

# Introduction

***Please use the Assignments/bike-computer directory for this assignment.***

It’s your first day at work for a company called ‘Wahahahoe’. You are handed with an important assignment: re-structure a piece of already written code.

The code project is called ‘bike-computer’. It is a simulation program for a bike-computer. It will provide simulated values for the **measurement** of speed, cadence, heart-rate and power. It will **store** the measured values in memory and then do some **math** to provide current, minimum, average and values for the measured values. It will display these values when running the program in a terminal as can be seen in figure 1.1.



*Figure 1.1: running the ‘bike-tool’ project in a terminal displays the ‘bike-computer’ values.*

The code that you received consists of several methods that are placed in one file called ‘main.c’. After browsing through the code you will find many things in the code that may not yet be familiar to you.

|  |  |
| --- | --- |
| For instance, the use of enums: | Or the use of a struct: |

Or the use of arrays:­



But it’s the first day at the office, and after mentioning this to your colleague he says:

‘Don’t worry about it! Just ignore these constructs for now.

You will learn all those new things over the next weeks.’

# Assignment

Your assignment for today is to reorganize the code so it will be organized in separate modules using header files. Your colleague stands up and draws an overview of the modules for you on the whiteboard:

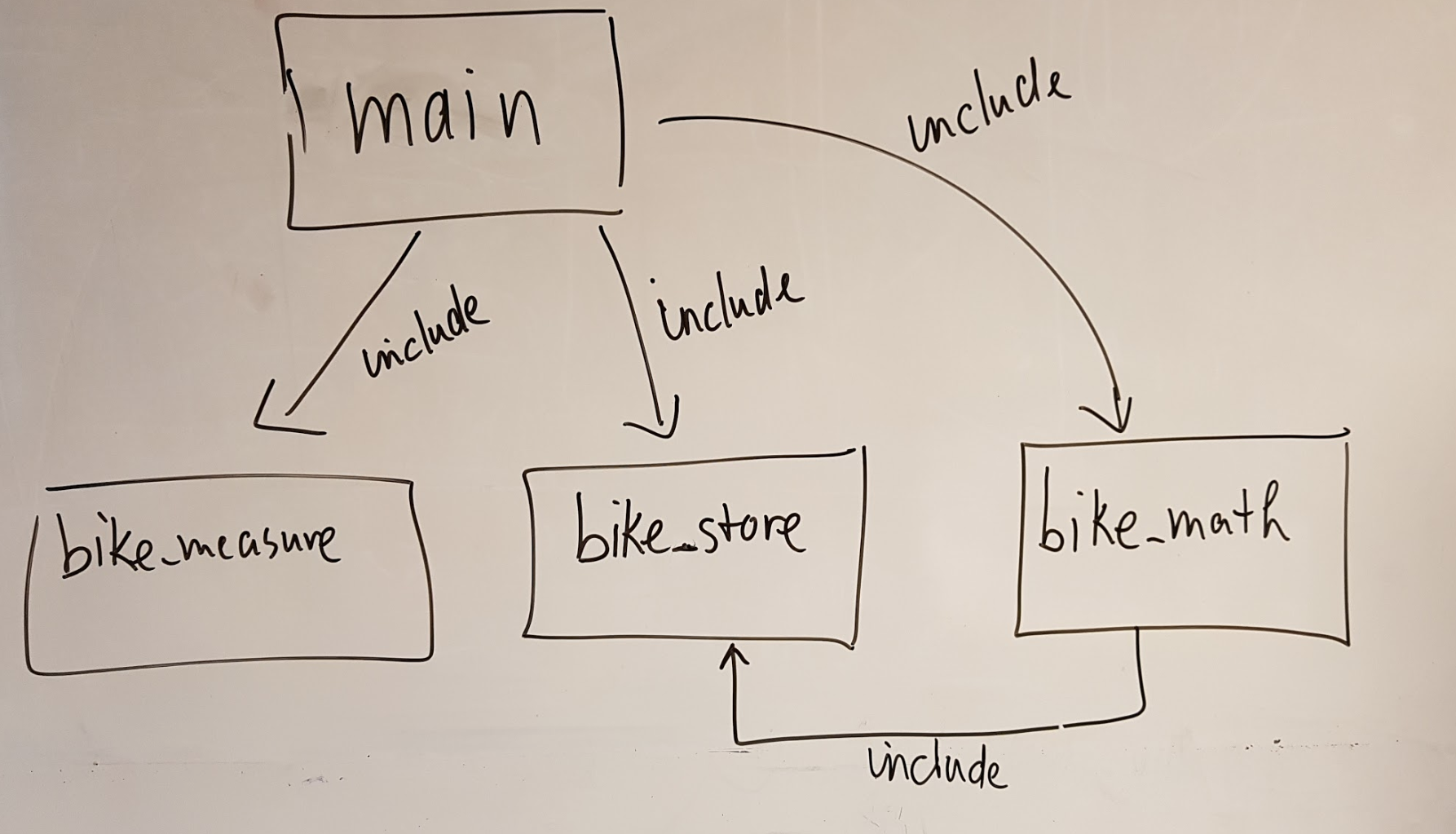
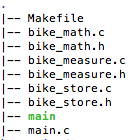


Figure 2.1: overview of the modules.

For each of the modules you must create two files:

* a **h**eader file (for instance: bike\_measure**.h**)
* a file that holds the functions' source code (for instance: bike\_measure**.c**)

At the end of the day you are expected to deliver the following files:



*Figure 2.2: overview of the files you are expected to deliver.*

You are expected to use common sense and **move** parts of the code from main.c to the separate modules. In order to do so, you will have to create a header file for each module.

# Possible questions

Your colleague says: ‘Anymore questions?’

You reply and ask the following questions:

1. How can I build and execute the existing code?
2. What is a header file, and how do I create it?
3. How do I get to use a method that is moved to one of the modules?
4. Why do I have to use these ‘#ifndef’ lines in the header files?
5. I have created the modules' files, but I get compiler errors when I try to run the project?
6. I get a warning about an unknown ‘uint16\_t’ for one of the header files?

Your colleague returns a friendly smile, and gives the following short answers to these questions:

1. You open up a terminal, navigate to the ‘bike-computer’ directory, then execute ‘make’ and then execute ‘./main’
2. Try to check out the ‘hello world’ example project that you can find in ‘<https://git.fhict.nl/technology/t-sem2-code/tree/master/es2/StartPointForExercises/Examples/helloworld>’. (You can also look in the StartPointForExercises' 'example' directory). Do check out ‘hello.h’!

1. You should for instance use: #include “bike-measure.h” in the main.c file.
2. This is something you can easily find on stackoverflow.
3. Are you sure you checked out the helloworld project ‘Makefile’? Check out how the file ‘hello.c’ is used. Likewise you should add all 3 bike-modules source files to the Makefile.
4. You have to ‘#include <stdint.h>. Make sure to also add this include to your header files when ‘uint16\_t’ is used. Again: see ‘hello.h’ in the helloworld example project.

Your colleague says: ‘good luck!’ and leaves. You will have to do some research on how things work. Luckily you can ask questions to the people of ‘Wahahahoe’ around you.

## Tips

* First try to move the methods that belong in the bike\_measure module. Create the c- and h-file for the bike\_measure module and try to get the project running with a ‘main.c’ and bike\_measure module.
* Figure 2.1 shows that:
  + the main.c file has to #include the bike\_measure, bike\_store and bike\_math module.
  + the bike\_math module has to #include the bike\_store module.

## Deliverables

You are expected to reorganize the project over the given modules, using header files, and adjusted Makefile and deliver the files as show in figure 2.2.