Study the video of the next lesson entirely:

Classes, objects and modules

Experiment independently with the example programs covered therein. A link to their source code is below the video. You can ask questions about them in the next lesson.

Excerpts from the current lesson:

To be turned in no later than 2 days before the next class.

The program has a prescribed structure. Improvements to it are allowed in consultation with the instructor. Start your source file again with requirements, test specs and design.

Don't work with classes and methods yet. Indeed, the purpose of this task is to show how small actually is the step from function-structured languages like C to object-oriented languages like C++.

Create a program that declares a struct type *Vec3D*. Variables of this type are vectors with 3 components. Such a declaration looks like this in C++:

```
struct Vec3D {float x, y, z;};
```

In doing so, make the following functions:

A function with signature `Vec3D vec3D (float x, float y, float z)` that creates, initializes and returns variables of this type.

A function with signature `void show (st::string label, Vec3D const & self)` that prints variables of this type in a console window, labeled with their name and terminated with a newline.

A function with signature `void show (st::string label, float scalar)` that prints floating point scalars in a console window, labeled with their name and terminated with a newline.

A function with signature `void show ()` that prints only a newline.

A function with signature `Vec3D minus (Vec3D const & self)` that returns a vector pointing the other way as *self*.

A function with signature `Vec3D add (Vec3D const & self, Vec3D const & other)` that returns the sum of *self* and *other*.

A function with signature `Vec3D sub (Vec3D const & self, Vec3D const & other)` that returns the difference of *self* and *other*.

A function with signature `Vec3D mul (Vec3D const & self, float scalar)` that returns the product of *self* and *scalar*.

A function with signature `Vec3D div (Vec3D const & self, float scalar)` that returns the quotient of *self* and *scalar*.

A function with signature float norm (Vec3D const & self) that returns the norm (length) of *self*.

A function with signature Vec3D unit (Vec3D const & self) that returns a vector with the same direction as its *self*, but with length 1.

A function with signature float dot (Vec3D const & self, Vec3D const & other) that returns the inproduct (dot product) of *self* and *other*.

A function with signature Vec3D cross (Vec3D const & self, Vec3D const & other) that returns the out product (cross product) of *self* and *other*.

The main function with signature int main () that tests all these functions in accordance with the test specs you have set up.

This task is another step toward the final problem. Terms such as *sum, difference, product* and *quotient* are explained in the *Numbers* lesson of the *Applied Mathematics* module. The other math required is explained in the *Equations, Matrices and Vectors* lesson from that same module, including the terms *vector*, *inproduct* and *out product*. The same issues are also covered in the Linear Algebra course. However, you can study the videos independently.

The assessment of this homework assignment is part of your final grade. What is assessed is not just a working or not working program, but more importantly whether you have understood both in broad outline and in detail what you are doing and can justify the choices made. Program and sensibly dosed commentary again in English.

--//--