Course C^{++}, Exercise 2

Deadline: 12 March 2013

Topic of this exercise is the general structure of a C++ program. You have to create a program that consists of different files, and construct a suitable Makefile for it. Make a separate directory for the task.

1. Download the files **rational.h**, **rational.cpp**, **vector.h**, **vector.cpp** and **main.cpp** from `http://www.ii.uni.wroc.pl/~nivelle/teaching/cpp2013/`.

2. The usual structure of a C^{++} program is as follows: Each class has two files, a file **class.h** and a file **class.cpp**. The **.h** file contains the declarations of the fields, and the declarations of the methods. The **.cpp** file contains the actual implementations of the methods.

   My experience is that nearly all code belongs to some class. If some code does not belong to a class, you should still make two files for it. Don't invent an artificial class name.

   The **.cpp** files are compiled separately. This is important in big projects, so we practice this from the beginning. Each **.cpp** file is compiled separately into a **.o** file. When all **.o** files are created, they are combined into a single, executable file. This process is called *linking.* The task of the linker is to check for identifiers that occur in one of the **.o** files, and that are defined in another file. When such identifiers exist, they are replaced by the address reference. If the linker cannot find a definition, it will produce an (generally incomprehensible) error message.

   Construct a Makefile that compiles the files above, using separate compilation. The main file contains some C^{++}-11 syntax, so you have to give the option `-std=c`+0x + to the compiler.

   When the Makefile is finished, you get a wall of linker errors. We deal with this in the next task. Still it should be possible to type `make vector.o` and compile the vector.

3. The linker errors are caused by methods that are declared in **rational.h**, but not defined in **rational.cpp**. Complete the missing functions.

4. Now it should be possible to run the complete program. Compute

$$\begin{pmatrix} \frac{1}{2} \\ \frac{3}{4} \\ \frac{1}{7} \end{pmatrix} \times \begin{pmatrix} \frac{1}{7} \\ \frac{1}{8} \\ \frac{-1}{4} \end{pmatrix}.$$

Verify by example that dotproduct and crossproduct are distributive:

$$\overline{v}_1 \times (\overline{w}_1 + \overline{w}_2) = \overline{v}_1 \times \overline{w}_1 + \overline{v}_1 \times \overline{w}_2.$$

$$\overline{v}_1.(\overline{w}_1 + \overline{w}_2) = \overline{v}_1.\overline{w}_1 + \overline{v}_1.\overline{w}_2.$$

You may test other laws as well, like for example associativity. Since the **vector** is in a namespace **linalg**, you need to to use the full names for its members: `linalg::vector vect = linalg::crossproduct( .... );`