

Exercises for the Lecture Series
“Object-Oriented Programming for Scientific Computing”

Ole Klein
ole.klein@iwr.uni-heidelberg.de

To be handed in on 07. 07. 2015 before the lecture

EXERCISE 1 FIBONNACI NUMBERS

In 1202 the mathematician Leonardo Fibonacci studied rabbit populations. He set up the following gedankenexperiment:



A newborn couple of rabbits is placed in an enclosed garden. Each pair of rabbits produces during its lifetime another pair of rabbits every month. A newborn pair is fertile after one month and therefore has its first offspring after two months. Assume for simplicity that rabbits don't die. How many rabbit couples are there in the garden after one year?

Today this is known as exponential growth. The number of rabbits in the pen is described by the Fibonacci sequence:

$$\begin{aligned}F_0 &= 1 \\F_1 &= 1 \\F_n &= F_{n-1} + F_{n-2}\end{aligned}$$

- Program the sequence as a template metaprogram. `Fibonacci<n>::value` has to provide the n-th number in the sequence.
- Also program the sequence using `constexpr`.
- Additionally program a function `int fibonacci(int n)` which dynamically calculates the n-th number using the recursion above.
- Measure the duration for both the compilation and the calculation in all three cases for growing `n`. Are there differences between the two static versions? And what is the tradeoff in comparison to the dynamic version? What complexity in time ($T(n) = \mathcal{O}(?)$ in Landau notation) can you observe both in terms of translation time and runtime?

8 Points

EXERCISE 2 CALCULATION WITH UNITS

The lecture explained how automatic calculation with units can be accomplished using templates. The used units were *m* (meters), *kg* (kilograms) and *s* (seconds). The space spanned by these three

basic units is called the MKS system. In this system most physical units can be represented, with the exception of the units of electrodynamics.

Implement the following two extensions:

- The basic units are expanded by the ampere A (unit of electric current), analogous to the SI system. Through the addition of this unit the complete electrodynamics can be described. Accordingly we call this the MKSA system.

Suppose you would add additional SI units, including the coulomb C (unit of electric charge). What problem would this cause? Have a look at the watt W for this, the unit of power. How can this be prevented?

- The units are expressed as follows in cm, g and s :

$$C \equiv 10^{-1} cm^{0.5} g^{0.5}$$
$$A \equiv 10^{-1} cm^{0.5} g^{0.5} s^{-1}$$

This is one of the possible extensions of the CGS system, as they are customary in physics. In this case, no extension of the basic units is necessary, but non-integer exponents for the units are needed. How can this be implemented, considering that floating point numbers can not be used as template parameters?

There are areas of physics where it is common to use completely different unit systems, e.g. because the considered quantities are unwieldy in the usual units or even because it isn't possible to store them in a `float`. Examples are the atomic units

$a_0 \approx 0.529 \cdot 10^{-10} m$	(Bohr atomic radius)
$u \approx 1.661 \cdot 10^{-27} kg$	(Atomic mass unit)
$e \approx 1.602 \cdot 10^{-19} C$	(Elementary charge)

and the plank units of quantum mechanics

$m_P \approx 2.177 \cdot 10^{-8} kg$	(Planck mass)
$l_P \approx 1.616 \cdot 10^{-35} m$	(Planck length)
$t_P \approx 5.391 \cdot 10^{-44} s$	(Planck time)

There are also the units of astronomy and in the non-scientific context the Anglo-American units (inch, foot, yard, mile...).

For efficiency reasons one wants to calculate with `float` and `double` in a fixed system of appropriate units and use the other systems at most for input and output. The same thing happens when the program is designed to support both metric and Anglo-American units or various scientific conventions simultaneously.

Let `Unit0` and `Unit1` be two systems as described in the lecture. Write a static infrastructure that allows the automatic conversion between the two systems, using the tools that were introduced in the lecture. You can assume that the two systems use the same three physical dimensions (mass, length, time) and that they are in the same order in both systems. This way only constants for rescaling are needed, but there is no mixing of the dimensions. A more general conversion infrastructure would of course need to be able to work also with systems that use completely different base dimensions, like the ones used in the first part of this exercise.

12 Points