ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programming Techniques for Scientific Simulations Exercise 4

HS 15
Prof. M. Troyer

Problem 4.1 Machine epsilon – revisited

Rewrite the program computing machine epsilon (Problem 1.5), but this time using templates. Compare your results against the standard numeric limits library (in the `<limits>` header) for types `float`, `double` and `long double`.

Problem 4.2 Penna model – part 1

Read the paper by Penna [T.J.P. Penna, J. Stat. Phys. 78, 1629 (1995)] (you may find it on the lecture homepage). Think over the structure for a Penna model simulation. Summarize the stated concepts. What are the features which all individuals have in common? Which features are different? How would you represent an individual in your code? Do not write any code yet.

Problem 4.3 Random and timer libraries

In the lecture repository you find a benchmark program, `benchmark/main.cpp`, for a random number generator using a `timer` and a `random` library that are not provided. Your task is to code and install them, so that the benchmark program works.

a) **`Timer` library**
Wrap the timing functions you used in the previous exercise in a simple class.

Use CMake to build and install it as a library. Note that the interface of the timer class has to be consistent with the one used in the benchmark program.

b) **`Random` library**
Implement a linear congruential random number generator:

$$X_{n+1} = (aX_n + c) \mod m, \tag{1}$$

where m is the largest number that your generator will output. X_n, a and c are integers and $m > 0$, $0 < a < m$ and $0 \leq c < m$ are constants. X_0 is the initial seed. You may use the following set of constants:

$$\begin{aligned} m &= 2^{32} \\ a &= 1664525 \\ c &= 1013904223 \end{aligned}$$

Use CMake to build and install the random number generator as a library. Note that the interface of the generator class has to be consistent with the one used in the benchmark program.

c) **Benchmark the random number generator**
Use CMake to compile the benchmark application. The configuration script should look for the `timer` and `random` libraries in their installation directories.

d) **Cross-platform compatibility**
Team up with a partner, preferably one with a machine running a different OS, and test if each others codes work on both machines.