ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Programming Techniques for Scientific Simulations

HS 2023
Dr. R. Käppeli,
V. Armegioiu,
I. Méndez Veiga, M. Herde,
F. Sasseli, N. Siegenheim

Exercise sheet 7

Due date: Monday, November 6, 2023, 23:59 CET (midnight).

Problem 7.1 Iterators

As we will see, iterators provide the interface between containers and algorithms, and thus play a key role in most scientific programs. In this exercise, we will provide a `MyIterator` class for the `SArray<T>` class discussed during the lecture (prototype files `my_iterator.hpp` and `sarray.hpp` are provided). Your goals for today are:

1. Implement a *forward iterator* class.
2. Make it a checked iterator, i.e. prevent illegal operations like `++(array.end())`.
3. (Bonus) Upgrade your forward iterator to a *bidirectional iterator*.

Hints:

- Make use of `iterators/my_iterator.hpp` for a skeleton of the code and the precise requirements on forward and bidirectional iterators.
- The `check()` method of `MyIterator` needs to verify that the `SArray<T>` entry currently pointed by `MyIterator` lies in the closed interval [`first_entry`, `last_entry`]. `MyIterator` can include private members that express this interval.
- Make sure to implement member functions `begin()` and `end()` in `iterators/sarray.hpp` for the `SArray<T>` class so that you may easily obtain iterators for your arrays.
- File `iterators/main.cpp` contains a simple program with which you may test some functionalities of your iterator.

Problem 7.2 Benchmarking Standard Containers

We want to benchmark the time required to front insert and erase in the three types of containers provided by the standard library: `std::vector`, `std::list` and `std::set`.

In order to achieve this, you should:

1. Create an array/vector of size n and assign its entries with a strictly monotone function of their index, for example `array[i] = i + 1`.
2. Copy the content of the array/vector into a `std::vector`, `std::list` and `std::set`.
3. For each container record the time to:
   - Insert a new element with value `0` at the beginning of the container (with method `insert`).
   - Undo the previous operation by erasing the inserted element (with method `erase`).

Perform benchmarks for many system sizes n (suggested: 2^4–2^{14}) and plot n vs. time on a log-log scale.

Hints:

- You are encouraged to write a single template function for benchmarking the three containers.
- Since `std::set` does not hold duplicates, make sure that you do not have the value `0` in the input array/vector, so that you can insert a new element with value `0` as per task 3. Take care that the size of the container does not shrink during the benchmark.
- To measure the time, use the library `Timer` from Problem 6.3!
- For a proper measurement, you should get the time between k repetitions of step 3, where $k \approx 1\,000\,000$.
- Some sample plotting scripts are provided (gnuplot `containers.gnuplot` or with Python/Matplotlib `plot.py`).
- When using `CMake`, perform the benchmarks using the `Release` build type. The build type is controlled via the variable `CMAKE_BUILD_TYPE`
  (i.e., use `cmake -DCMAKE_BUILD_TYPE=Release ..`).

Problem 7.3 Penna Model Implementation

Design and implement a `Population` class that performs all major operations on a population of animals (aging, deaths, generation of offsprings) and combine the classes into a working simulation of the Penna model. As usual, you can base your code on the classes `Genome` and `Animal` from the lecture repository, but working on your own design might prove more rewarding.

1. Once the simulation is working, test it by setting the simulation parameters to similar values as in the paper ($M = 2, T = 2, R = 8, N_0 = 1\,000, N_{\max} = 10\,000$) and reproducing its Fig. 1: population number as a function of time, with $T_{\max} = 10\,000$ (which should also be the number of simulated measurements).
2. (Bonus) Plot the distribution of bad genes in a genome at the beginning and at the end of the simulation.