Time Duration

This lesson gives a brief introduction to a time duration class template and explains it with the help of interactive examples.

Time duration `std::chrono::duration` is a class template that consists of the type of the tick `Rep` and the length of a tick `Period`.

```
template<
    class Rep,
    class Period = std::ratio<1>
> class duration;
```

The tick length is `std::ratio<1>` by default; `std::ratio<1>` stands for a second and can also be written as `std::ratio<1,1>`. The rest is quite easy. `std::ratio<60>` is a minute and `std::ratio<1,1000>` a millisecond. When the type of `Rep` is a floating-point number, you can use it to hold fractions of time ticks.

C++11 predefines the most important time durations:

```
typedef duration<signed int, nano> nanoseconds;
typedef duration<signed int, micro> microseconds;
typedef duration<signed int, milli> milliseconds;
typedef duration<signed int> seconds;
typedef duration<signed int, ratio< 60>> minutes;
typedef duration<signed int, ratio<3600>> hours;
```

How much time has passed since the UNIX epoch (1.1.1970)? Thanks to type aliases for the different time durations, I can answer the question quite easily. In the following example, I ignore leap years and assume that a year has 365 days.

```
// timeSinceEpoch.cpp

#include <chrono>
#include <iostream>
```

```
using namespace std;

int main(){


  cout << fixed << endl;

  cout << "Time since 1.1.1970:\n" << endl;

  const auto timeNow= chrono::system_clock::now();
  const auto duration= timeNow.time_since_epoch();
  cout << duration.count() << " nanoseconds " << endl;

  typedef chrono::duration<long double, ratio<1, 1000000>> MyMicroSecondTick;
  MyMicroSecondTick micro(duration);
  cout << micro.count() << " microseconds" << endl;

  typedef chrono::duration<long double, ratio<1, 1000>> MyMilliSecondTick;
  MyMilliSecondTick milli(duration);
  cout << milli.count() << " milliseconds" << endl;

  typedef chrono::duration<long double> MySecondTick;
  MySecondTick sec(duration);
  cout << sec.count() << " seconds " << endl;

  typedef chrono::duration<double, ratio<60>> MyMinuteTick;
  MyMinuteTick myMinute(duration);
  cout << myMinute.count() << " minutes" << endl;

  typedef chrono::duration<double, ratio<60*60>> MyHourTick;
  MyHourTick myHour(duration);
  cout << myHour.count() << " hours" << endl;

  typedef chrono::duration<double, ratio<60*60*24*365>> MyYearTick;
  MyYearTick myYear(duration);
  cout << myYear.count() << " years" << endl;

  typedef chrono::duration<double, ratio<60*45>> MyLessonTick;
  MyLessonTick myLesson(duration);
  cout << myLesson.count() << " lessons" << endl;

  cout << endl;

}
```

The typical time durations are microsecond (line 18), millisecond (line 22), second (line 26), minute (line 30), hour (line 34), and year (line 38). Also, I define the German school hour (45 min) in line 42.

As the next lesson illustrates, it's quite convenient to calculate with time durations.