p0647r0 - Floating point value access for std::ratio

Peter Sommerlad
2017-05-20

<table>
<thead>
<tr>
<th>Document Number:</th>
<th>p0647r0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>2017-05-20</td>
</tr>
<tr>
<td>Project:</td>
<td>Programming Language C++</td>
</tr>
<tr>
<td>Audience:</td>
<td>LWG/LEWG</td>
</tr>
<tr>
<td>Target:</td>
<td>C++20</td>
</tr>
</tbody>
</table>

1 Motivation

Preparing for standardizing units and using std::ratio for keeping track of fractions often one needs to get the quotient as a floating point number or as a number of a type underlying a quantity, e.g., a fixpoint type. Doing that manually means adding a cast before doing the division. This is tedious and it would be nice to just access the value, as one can do with std::integral_constant. I believe that omission is just a historical accident, because it was not possible to do compile-time computation with floating point values when ratio was invented. There are some options on how to access the fraction as a compile-time entity. I chose to make the value member a long double and provide a templatized explicit conversion operator for accessing the fraction.

2 Acknowledgements

— Authors of N2661: Howard Hinnant, Walter Brown, Jeff Garland, Marc Paterno.
— Members of the LiaW workshop 'Towards Units' at C++Now 2017: Billy Baker, Charles Wilson, Daniel Pfeifer, Dave Jenkins, Manuel Bergler, Morris Hafner, Nicolas Holthaus, Peter Bindels, Steven Watanabe, Tuan Tran.
3 Changes Proposed

Modify section 23.26.3 by inserting floating point access to the fractional value represented.

3.0.1 Class template ratio

namespace std {
    template <intmax_t N, intmax_t D = 1>
    class ratio {
        public:
            static constexpr intmax_t num;
            static constexpr intmax_t den;

            using type = ratio<num, den>;

            static constexpr long double value{ static_cast<long double>(num)/den };
            template<typename R>
            explicit constexpr operator R() const noexcept {
                return static_cast<R>(num)/static_cast<R>(den);
            }
    };
}