Clarify intent of P1841 numeric traits

ABSTRACT

A list of design-related questions after implementation of [P1841R2] “Wording for Individually Specializable Numeric Traits”.

CONTENTS

1 Introduction 1
2 Design Questions 1
3 Suggested Straw Polls 2
A Bibliography 3
1 **INTRODUCTION**

[P1841R2] provides wording for numeric traits. The last design paper was [P0437R1] with additions from [P1370R1].

2 **DESIGN QUESTIONS**

1. When exactly is a trait disabled for a given numeric type? It seems the intent was for the value member to be defined whenever a representation for the desired constant exists. The wording needs to clarify whether any behavioral aspects play a role. For example, a denorm_min may be enabled independent of whether the execution environment flushes denormals to zero / treats denormals as zero. Even in the case of a processor that unconditionally zeros denormals; as long as a representation exists, is the trait enabled? Conversely, if a representation does not exist, is the trait disabled? Specifically, denorm_min should never have the value of norm_min?

2. Please clarify whether we want to treat bool as a numeric type and enable the traits accordingly. The current wording in [P1841R2] enables the traits for bool, which is consistent with std::numeric_limits. std::numeric_limits<bool> will still exist if needed. Numeric code does not use bool as a numeric type, despite it being technically an “arithmetic type” in the core language.

3. Many of the numeric traits are motivated by floating-point and make little sense for integral types. Is it intended that all of the following numeric traits are enabled also for integral types?
   - denorm_min
   - epsilon
   - norm_min
   - reciprocal_overflow_threshold
   - round_error
   - max_exponent
   - max_exponent10
   - min_exponent
   - min_exponent10
4. reciprocal_overflow_threshold yields a subnormal number for IEC559 types. How should this value change wrt. treat-denormals-as-zero? I.e. in a situation where the hardware treats subnormal operands as zero you get $1/0 \to \text{inf}$, which does overflow. In which case it doesn't match the specification anymore (“The smallest positive value $x$ of type $T$ such that $T(1)/x$ does not overflow”). This trait is specified by a behavior and as such may depend on processor state. As a compile-time constant this value must be independent from runtime behavior. But what is the correct value?

5. numeric_limits::max_digits10 is 0 for integral types. Is max_digits10_v<int> supposed to yield digits10_v<int> + 1? Or should it only be specialized for floating-point?

### SUGGESTED STRAW POLLS

Poll: Whether a numeric trait is enabled is independent of processor behavior and only reflects whether a representation for the requested trait exists (ignoring reciprocal_overflow_threshold).

<table>
<thead>
<tr>
<th>SF</th>
<th>F</th>
<th>N</th>
<th>A</th>
<th>SA</th>
</tr>
</thead>
</table>

Poll: All numeric traits for bool should be disabled.

<table>
<thead>
<tr>
<th>SF</th>
<th>F</th>
<th>N</th>
<th>A</th>
<th>SA</th>
</tr>
</thead>
</table>

Poll: The numeric traits listed in item 3 in P2551R0 should be disabled for integral types.

<table>
<thead>
<tr>
<th>SF</th>
<th>F</th>
<th>N</th>
<th>A</th>
<th>SA</th>
</tr>
</thead>
</table>

Poll: reciprocal_overflow_threshold should be independent of processor behavior and only reflect the value range of possible representations of the given type.

<table>
<thead>
<tr>
<th>SF</th>
<th>F</th>
<th>N</th>
<th>A</th>
<th>SA</th>
</tr>
</thead>
</table>

Poll: reciprocal_overflow_threshold should reflect processor behavior if it is known at compile-time (e.g. the target hardware unconditionally treats denormals as zero), otherwise it should reflect the value range of possible representations of the given type.

<table>
<thead>
<tr>
<th>SF</th>
<th>F</th>
<th>N</th>
<th>A</th>
<th>SA</th>
</tr>
</thead>
</table>
Poll: max_digits10 should deviate from numeric_limits and yields digits10_v<T> + 1.

A BIBLIOGRAPHY

