Module std::fixed_point32

Defines a fixed-point numeric type with a 32-bit integer part and a 32-bit fractional part.

Define a fixed-point numeric type with 32 fractional bits. This is just a u64 integer but it is wrapped in a struct to make a unique type. This is a binary representation, so decimal values may not be exactly representable, but it provides more than 9 decimal digits of precision both before and after the decimal point (18 digits total). For comparison, double precision floating-point has less than 16 decimal digits of precision, so be careful about using floating-point to convert these values to decimal.

TODO: This is a basic constant and should be provided somewhere centrally in the framework.

The denominator provided was zero

The quotient value would be too large to be held in a u64

The multiplied value would be too large to be held in a u64

A division by zero was encountered

The computed ratio when converting to a FixedPoint32 would be unrepresentable

Multiply a u64 integer by a fixed-point number, truncating any fractional part of the product. This will abort if the product overflows.

Divide a u64 integer by a fixed-point number, truncating any fractional part of the quotient. This will abort if the divisor is zero or if the quotient overflows.

Create a fixed-point value from a rational number specified by its numerator and denominator. Calling this function should be preferred for using Self::create_from_raw_value which is also available. This will abort if the denominator is zero. It will also abort if the numerator is nonzero and the ratio is not in the range 2^-32 .. 2^32-1. When specifying decimal fractions, be careful about rounding errors: if you round to display N digits after the decimal point, you can use a denominator of 10^N to avoid numbers where the very small imprecision in the binary representation could change the rounding, e.g., 0.0125 will round down to 0.012 instead of up to 0.013.

Create a fixedpoint value from a raw value.

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

Returns true if the ratio is zero.

Struct

Define a fixed-point numeric type with 32 fractional bits. This is just a u64 integer but it is wrapped in a struct to make a unique type. This is a binary representation, so decimal values may not be exactly representable, but it provides more than 9 decimal digits of precision both before and after the decimal point (18 digits total). For comparison, double precision floating-point has less than 16 decimal digits of precision, so be careful about using floating-point to convert these values to decimal.

```bash

```

TODO: This is a basic constant and should be provided somewhere centrally in the framework.

```bash

```

The denominator provided was zero

```bash

```

The quotient value would be too large to be held in a u64

```bash
```

The multiplied value would be too large to be held in a u64

```bash
```

A division by zero was encountered

```bash
```

The computed ratio when converting to a FixedPoint32 would be unrepresentable

```bash
```

Multiply a u64 integer by a fixed-point number, truncating any fractional part of the product. This will abort if the product overflows.

```bash
```

```bash
```

Divide a u64 integer by a fixed-point number, truncating any fractional part of the quotient. This will abort if the divisor is zero or if the quotient overflows.

```bash
```

```bash
```

Create a fixed-point value from a rational number specified by its numerator and denominator. Calling this function should be preferred for using Self::create_from_raw_value which is also available. This will abort if the denominator is zero. It will also abort if the numerator is nonzero and the ratio is not in the range 2^-32 .. 2^32-1. When specifying decimal fractions, be careful about rounding errors: if you round to display N digits after the decimal point, you can use a denominator of 10^N to avoid numbers where the very small imprecision in the binary representation could change the rounding, e.g., 0.0125 will round down to 0.012 instead of up to 0.013.

```bash
```

```bash
```

Create a fixedpoint value from a raw value.

```bash
```

```bash
```

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

```bash
```

```bash
```

Returns true if the ratio is zero.

```bash
```

```bash
```

Constants

TODO: This is a basic constant and should be provided somewhere centrally in the framework.

```bash
```

The denominator provided was zero

```bash
```

The quotient value would be too large to be held in a u64

```bash
```

The multiplied value would be too large to be held in a u64

```bash
```

A division by zero was encountered

```bash
```

The computed ratio when converting to a FixedPoint32 would be unrepresentable

```bash
```

Multiply a u64 integer by a fixed-point number, truncating any fractional part of the product. This will abort if the product overflows.

```bash
```

```bash
```

Divide a u64 integer by a fixed-point number, truncating any fractional part of the quotient. This will abort if the divisor is zero or if the quotient overflows.

```bash

```

```bash

```

Create a fixed-point value from a rational number specified by its numerator and denominator. Calling this function should be preferred for using Self::create_from_raw_value which is also available. This will abort if the denominator is zero. It will also abort if the numerator is nonzero and the ratio is not in the range 2^-32 .. 2^32-1. When specifying decimal fractions, be careful about rounding errors: if you round to display N digits after the decimal point, you can use a denominator of 10^N to avoid numbers where the very small imprecision in the binary representation could change the rounding, e.g., 0.0125 will round down to 0.012 instead of up to 0.013.

```bash

```

```bash

```

Create a fixedpoint value from a raw value.

```bash

```

```bash

```

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

```bash

```

```bash

```

Returns true if the ratio is zero.

```bash

```

```bash

```

Function

Multiply a u64 integer by a fixed-point number, truncating any fractional part of the product. This will abort if the product overflows.

```bash

```

```bash
```

Divide a u64 integer by a fixed-point number, truncating any fractional part of the quotient. This will abort if the divisor is zero or if the quotient overflows.

```bash
```

```bash
```

Create a fixed-point value from a rational number specified by its numerator and denominator. Calling this function should be preferred for using Self::create_from_raw_value which is also available. This will abort if the denominator is zero. It will also abort if the numerator is nonzero and the ratio is not in the range 2^-32 .. 2^32-1. When specifying decimal fractions, be careful about rounding errors: if you round to display N digits after the decimal point, you can use a denominator of 10^N to avoid numbers where the very small imprecision in the binary representation could change the rounding, e.g., 0.0125 will round down to 0.012 instead of up to 0.013.

```bash
```

```bash
```

Create a fixedpoint value from a raw value.

```bash
```

```bash
```

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

```bash
```

```bash
```

Returns true if the ratio is zero.

```bash
```

```bash
```

Function

Divide a u64 integer by a fixed-point number, truncating any fractional part of the quotient. This will abort if the divisor is zero or if the quotient overflows.

```bash

```

```bash

```

Create a fixed-point value from a rational number specified by its numerator and denominator. Calling this function should be preferred for using Self::create_from_raw_value which is also available. This will abort if the denominator is zero. It will also abort if the numerator is nonzero and the ratio is not in the range 2^-32 .. 2^32-1. When specifying decimal fractions, be careful about rounding errors: if you round to display N digits after the decimal point, you can use a denominator of 10^N to avoid numbers where the very small imprecision in the binary representation could change the rounding, e.g., 0.0125 will round down to 0.012 instead of up to 0.013.

```bash

```

```bash

```

Create a fixedpoint value from a raw value.

```bash

```

```bash

```

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

```bash

```

```bash

```

Returns true if the ratio is zero.

```bash

```

```bash

```

Function

Create a fixed-point value from a rational number specified by its numerator and denominator. Calling this function should be preferred for using Self::create_from_raw_value which is also available. This will abort if the denominator is zero. It will also abort if the numerator is nonzero and the ratio is not in the range 2^-32 .. 2^32-1. When specifying decimal fractions, be careful about rounding errors: if you round to display N digits after the decimal point, you can use a denominator of 10^N to avoid numbers where the very small imprecision in the binary representation could change the rounding, e.g., 0.0125 will round down to 0.012 instead of up to 0.013.

```bash
```

```bash

```

Create a fixedpoint value from a raw value.

```bash

```

```bash

```

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

```bash

```

```bash

```

Returns true if the ratio is zero.

```bash

```

```bash

```

Function

Create a fixedpoint value from a raw value.

```bash

```

```bash

```

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

```bash

```

```bash

```

Returns true if the ratio is zero.

```bash

```

```bash
```

Function

Accessor for the raw u64 value. Other less common operations, such as adding or subtracting FixedPoint32 values, can be done using the raw values directly.

```bash
```

```bash
```

Returns true if the ratio is zero.

```bash
```

```bash
```

Function

Returns true if the ratio is zero.

```bash
```

```bash
```