Proposal for C2X
WG14 N2532

Title:	Min-max functions
Author, affiliation:	C FP group
Date:	2020-05-15
Proposal category:	Technical
Reference:	N2489, N2478, N2531

This proposal is essentially the same as N2489 but is updated to refer to newly published IEC 60559:2020.

Problem:

The 2020 revision of IEC 60559 (IEEE 754:2019) replaces the minNum, maxNum, minNumMag, and maxNumMag operations from IEC 60559:2011 (IEEE 754:2008), which are supported by the **`fmin`**, **`fmax`** **`fminmag`**, and **`fmaxmag`** functions in draft C2X (N2478), with two sets of new min-max operations:

minimum, maximum, miniumMag, maximumMag

and

minimumNum, maximumNum, minimumMagNum, maximumMagNum

These two sets differ primarily in their treatment of NaN operands. None of the new operations is equivalent to a min-max operation in IEC 60559:2011.

The floating-point committee took this unusual step because it believed the min-max operations IEC 60559:2011 were seriously flawed and problematic for their intended use. The new min-max operations are optional in IEC 60559:2020, but only because the revision was restricted from adding requirements. The next revision is expected to make the new operations mandatory.

The suggested changes below update draft C2X to support the new min-max operations in IEC 60559:2020 as new **`<math.h>`** functions. Also, given the floating-point committee's thinking about the min-max operations in its previous standard, this proposal removes the **`fminmag`** and **`fmaxmag`** functions, which were just recently introduced into draft C2X via TS 18661-1. This proposal does not remove the **`fmin`** and **`fmax`** functions, for compatibility reasons.

See also N2531 regarding C support for IEC 60559:2020.

Suggested changes

In the table "Preferred quantum exponents" in 5.2.4.2.3#7, replace:

`fmin`, **`fmax`**, **`fminmag`**, **`fmaxmag`**

with

`fmin`, **`fmax`**, **`fminimum`**, **`fmaximum`**, **`fminimum_mag`**, **`fmaximum_mag`**, **`fminimum_num`**, **`fmaximum_num`**, **`fminimum_mag_num`**, **`fmaximum_mag_num`**

In the table "Operation binding" in F.3, replace:

minNum	**`fmin`**	7.12.12.3, F.10.9.3
maxNum	**`fmax`**	7.12.12.2, F.10.9.2
minNumMag	**`fminmag`**	7.12.12.5, F.10.9.5
maxNumMag	**`fmaxmag`**	7.12.12.4, F.10.9.4

with:

	`fmax`	7.12.12.2, F.10.9.2
	`fmin`	7.12.12.3, F.10.9.3
maximum	**`fmaximum`**	7.12.12.4, F.10.9.4
minimum	**`fminimum`**	7.12.12.5, F.10.9.4
maximumMagnitude	**`fmaximum_mag`**	7.12.12.6, F.10.9.4
minimumMagnitude	**`fminimum_mag`**	7.12.12.7, F.10.9.4
maximumNumber	**`fmaximum_num`**	7.12.12.8, F.10.9.5
minimumNumber	**`fminimum_num`**	7.12.12.9, F.10.9.5
maximumMagnitudeNumber	**`fmaximum_mag_num`**	7.12.12.10, F.10.9.5
minimumMagnitudeNumber	**`fminimum_mag_num`**	7.12.12.11, F.10.9.5

In F.3, after #2, insert:

[2a] The **`fmin`** and **`fmax`** functions provide the minNum and maxNum operations specified in IEC 60559:2011.

Remove 7.12.12.4 and 7.12.12.5 and all references to **`fmaxmag`** and **`fminmag`** functions.

After 7.12.12.3, add:

7.12.12.4 The `fmaximum` functions

Synopsis

[1]
```
#include <math.h>
double fmaximum(double x, double y);
float fmaximumf(float x, float y);
long double fmaximuml(long double x, long double y);
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fmaximumd32(_Decimal32 x, _Decimal32 y);
_Decimal64 fmaximumd64(_Decimal64 x, _Decimal64 y);
_Decimal128 fmaximumd128(_Decimal128 x, _Decimal128 y);
#endif
```

Description

[2] The **`fmaximum`** functions determine the maximum value of their arguments. For these functions, +0 is considered greater than −0. These functions differ from the **`fmaximum_num`** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **`fmaximum`** functions return the maximum value of their arguments.

7.12.12.5 The `fminimum` functions

Synopsis

[1]
```
#include <math.h>
double fminimum(double x, double y);
float fminimumf(float x, float y);
long double fminimuml(long double x, long double y);
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fminimumd32(_Decimal32 x, _Decimal32 y);
_Decimal64 fminimumd64(_Decimal64 x, _Decimal64 y);
_Decimal128 fminimumd128(_Decimal128 x, _Decimal128 y);
#endif
```

Description

[2] The **`fminimum`** functions determine the minimum value of their arguments. For these functions, −0 is considered less than +0. These functions differ from the **`fminimum_num`** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **`fminimum`** functions return the minimum value of their arguments.

7.12.12.6 The `fmaximum_mag` functions

Synopsis

[1]
```
#include <math.h>
double fmaximum_mag(double x, double y);
float fmaximum_magf(float x, float y);
long double fmaximum_magl(long double x, long double y);
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fmaximum_magd32(_Decimal32 x, _Decimal32 y);
_Decimal64 fmaximum_magd64(_Decimal64 x, _Decimal64 y);
_Decimal128 fmaximum_magd128(_Decimal128 x,
      _Decimal128 y);
#endif
```

Description

[2] The **fmaximum_mag** functions determine the value of the argument of maximum magnitude: **x** if |**x**| > |**y**|, **y** if |**y**| > |**x**|, and **fmaximum(x, y)** otherwise. These functions differ from the **fmaximum_mag_num** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fmaximum_mag** functions return the value of the argument of maximum magnitude.

7.12.12.7 The fminimum_mag functions

Synopsis

[1]
```
#include <math.h>
double fminimum_mag(double x, double y);
float fminimum_magf(float x, float y);
long double fminimum_magl(long double x, long double y);
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fminimum_magd32(_Decimal32 x, _Decimal32 y);
_Decimal64 fminimum_magd64(_Decimal64 x, _Decimal64 y);
_Decimal128 fminimum_magd128(_Decimal128 x,
      _Decimal128 y);
#endif
```

Description

[2] The **fminimum_mag** functions determine the value of the argument of minimum magnitude: **x** if |**x**| < |**y**|, **y** if |**y**| < |**x**|, and **fminimum(x, y)** otherwise. These functions differ from the **fminimum_mag_num** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fminimum_mag** functions return the value of the argument of minimum magnitude.

7.12.12.8 The fmaximum_num functions

Synopsis

[1]
```
#include <math.h>
double fmaximum_num(double x, double y);
float fmaximum_numf(float x, float y);
long double fmaximum_numl(long double x, long double y);
```

```
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fmaximum_numd32(_Decimal32 x, _Decimal32 y);
_Decimal64 fmaximum_numd64(_Decimal64 x, _Decimal64 y);
_Decimal128 fmaximum_numd128(_Decimal128 x,
      _Decimal128 y);
#endif
```

Description

[2] The **fmaximum_num** functions determine the maximum value of their numeric arguments. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **fmaximum** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fmaximum_num** functions return the maximum value of their numeric arguments.

7.12.12.9 The fminimum_num functions

Synopsis

[1]
```
#include <math.h>
double fminimum_num(double x, double y);
float fminimum_numf(float x, float y);
long double fminimum_numl(long double x, long double y);
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fminimum_numd32(_Decimal32 x, _Decimal32 y);
_Decimal64 fminimum_numd64(_Decimal64 x, _Decimal64 y);
_Decimal128 fminimum_numd128(_Decimal128 x,
      _Decimal128 y);
#endif
```

Description

[2] The **fminimum_num** functions determine the minimum value of their numeric arguments. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **fminimum** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **fminimum_num** functions return the minimum value of their numeric arguments.

7.12.12.10 The `fmaximum_mag_num` functions

Synopsis

[1]
```
#include <math.h>
double fmaximum_mag_num(double x, double y);
float fmaximum_mag_numf(float x, float y);
long double fmaximum_mag_numl(long double x,
      long double y);
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fmaximum_mag_numd32(_Decimal32 x,
      _Decimal32 y);
_Decimal64 fmaximum_mag_numd64(_Decimal64 x,
      _Decimal64 y);
_Decimal128 fmaximum_mag_numd128(_Decimal128 x,
      _Decimal128 y);
#endif
```

Description

[2] The **`fmaximum_mag_num`** functions determine the value of a numeric argument of maximum magnitude. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **`fmaximum_mag`** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **`fmaximum_mag_num`** functions return the value of a numeric argument of maximum magnitude.

7.12.12.11 The `fminimum_mag_num` functions

Synopsis

[1]
```
#include <math.h>
double fminimum_mag_num(double x, double y);
float fminimum_mag_numf(float x, float y);
long double fminimum_mag_numl(long double x,
      long double y);
#ifdef __STDC_IEC_60559_DFP__
_Decimal32 fminimum_mag_numd32(_Decimal32 x,
      _Decimal32 y);
_Decimal64 fminimum_mag_numd64(_Decimal64 x,
      _Decimal64 y);
_Decimal128 fminimum_mag_numd128(_Decimal128 x,
      _Decimal128 y);
#endif
```

Description

[2] The **`fminimum_mag_num`** functions determine the value of a numeric argument of minimum magnitude. They determine the number if one argument is a number and the other is a NaN. These functions differ from the **`fminimum_mag`** functions only in their treatment of NaN arguments (see F.10.9.4, F.10.9.5).

Returns

[5] The **`fminimum_mag_num`** functions return the value of a numeric argument of minimum magnitude.

NOTE The **`fmax`** and **`fmin`** functions are similar to the **`fmaximum_num`** and **`fminimum_num`** functions, though may differ in which signed zero is returned when the arguments are differently signed zeros and in their treatment of signaling NaNs (see F.10.9.5).

Remove F.10.9.4 and F.10.9.5.

After F.10.9.3 add:

F.10.9.4 The `fmaximum`, `fminimum`, `fmaximum_mag`, and `fminimum_mag` functions

These functions treat NaNs like other functions in **`<math.h>`** (see F.10). They differ from the corresponding **`fmaximum_num`**, **`fminimum_num`**, **`fmaximum_mag_num`**, and **`fminimum_mag_num`** functions only in their treatment of NaNs.

F.10.9.5 The `fmaximum_num`, `fminimum_num`, `fmaximum_mag_num`, and `fminimum_mag_num` functions

These functions return the number if one argument is a number and the other is a quiet or signaling NaN. If both arguments are NaNs, a quiet NaN is returned. If an argument is a signaling NaN, the “invalid” floating-point exception is raised (even though the function returns the number when the other argument is a number).

In F.2.1, change:

[4] Any operator or **`<math.h>`** function that raises an "invalid" floating-point exception, if delivering a floating type result, shall return a quiet NaN.

to:

[4] Any operator or **`<math.h>`** function that raises an "invalid" floating-point exception, if delivering a floating type result, shall return a quiet NaN, unless explicitly specified otherwise.

In 7.25#9, in the list of type-generic macros, remove: **`fminmag`** and **`fmaxmag`**.

In 7.25#9, in the list of type-generic macros, add: **`fmaximum`**, **`fminimum`**, **`fmaximum_num`**, **`fminimum_num`**, **`fmaximum_mag`**, **`fminimum_mag`**, **`fmaximum_mag_num`**, and **`fminimum_mag_num`**.