Proposal for C2Y
WG14 N3702

Title:	Standard pragmas and headers
Author, affiliation:	C FP group
Date:	2025-07-11
Proposal category:	Editorial
Reference:	N3550

Background

What is the nature of the association between a standard pragma (6.10.8) and the header whose library subclause specifies the pragma? Must the header be included before the pragma can be used? If so, what is the implementation to do if the pragma is used where the header has not been included?

Each standard pragma is specified in a library header subclause with a synopsis in the same form as the functions declared by the header. For example, the `CX_LIMITED_RANGE` pragma is specified in 7.3.4 as follows:

> **Synopsis**
>
> ```
> #include <complex.h>
> #pragma STDC CX_LIMITED_RANGE on-off-switch
> ```

This suggests the header must be included for the pragma to be available.

However, elsewhere the standard suggests by omission that header inclusion is not required. 7.1.2 #1 says

> Each library function is declared in a header, 221) whose contents are made available by the `#include` preprocessing directive. The header declares a set of related functions, plus any types and additional macros needed to facilitate the use of such related functions.

7.1.2 #5 says

> … If used, a header shall be included outside of any external declaration or definition, and it shall first be included before the first reference to any of the functions or objects it declares, or to any of the types or macros it defines.

The relevant library subclauses begin with a statement about what the header defines and declares, with no mention of pragmas. For example, 7.6.1 says

> The header `<fenv.h>` defines several macros, and declares types and functions that provide access to the floating-point environment.

It does not mention pragmas though 7.6 specifies three standard pragmas.

Use of a standard pragma does not require any of the definitions or declarations in the associated header. A *direction* indicated in an `FENV_ROUND` or `FENV_DEC_ROUND` pragma has the name of a rounding direction macro defined in the header, but this is just the name and it does not rely on the macro definition in the header.

All the standard pragmas have uses where the header inclusion is not needed.

For these reasons, this proposal supports, and clarifies, that use of a standard header does not require header inclusion.

A first approach was to move all the standard pragma specifications in the library to 6.10.8. It looked like that would be fine for `FP_CONTRACT`, but `FENV_ROUND` and `FENV_DEC_ROUND` have pages of semantics (including tables), some of it directly related to `<fenv.h>` features. The move would mean an excess of floating-point specification in 6.10.8 and extra cross-referencing for the reader.

Instead, the approach below leaves the descriptions of the standard pragmas in place, but it clearly states that use of a standard pragma does not require inclusion of the header whose subclause describes the pragma.

Suggested Changes

In 6.10.8 #2, change

> … whose meanings are described ~~elsewhere~~:

to

> … whose meanings are described in related library subclauses:

At the end of 6.10.8 #2, just before Recommended practice, add:

> Use of standard pragmas does not require inclusion of the header whose subclause contains the description of the pragma.

In 7.3.4 change

> ~~**Synapsis**~~
>
> ~~`#include <complex.h>`~~
> ~~`#pragma STDC CX_LIMITED_RANGE` *on-off-switch*~~
>
> ~~**Description**~~
> … The ~~`CX_LIMITED_RANGE`~~ pragma can be used to inform the implementation that (where the state is "on") the usual mathematical formulas are acceptable.[236] ~~-~~ …

to

... The standard pragma

```
#pragma STDC CX_LIMITED_RANGE on-off-switch
```

can be used to inform the implementation that (where the state is "on") the usual mathematical formulas are acceptable.[236] Use of the pragma does not require inclusion of <complex.h>. ...

In 7.6.2 change

~~**Synapsis**~~

```
~~#include <fenv.h>~~
~~#pragma STDC FENV_ACCESS on-off-switch~~
```

~~**Description**~~

... The ~~FENV_ACCESS~~ pragma provides a means to inform the implementation ~~when~~ a program can access the floating-point environment to test floating-point status flags or run under non-default floating-point control modes.[254] ...

to

... The standard pragma

```
#pragma STDC FENV_ACCESS on-off-switch
```

provides a means to inform the implementation where a program can access the floating-point environment to test floating-point status flags or run under non-default floating-point control modes.[254] Use of the pragma does not require inclusion of <fenv.h>. ...

In 7.6.3 change

~~**Synapsis**~~

```
~~#include <fenv.h>~~
~~#pragma STDC FENV_ROUND direction~~
~~#pragma STDC FENV_ACCESS FE_DYNAMIC~~
```

~~**Description**~~

... The ~~FENV_ROUND~~ pragma provides a means to specify a constant rounding direction for floating-point operations for standard floating types within a translation unit or compound statement. ...

to

... The standard pragma

```
#pragma STDC FENV_ROUND direction
```

provides a means to specify a constant rounding direction for floating-point operations for standard floating types within a translation unit or compound statement. Use of the pragma does not require inclusion of <fenv.h>. ...

In 7.6.4 change

~~Synapsis~~

```
~~#include <fenv.h>~~
~~#ifdef __STDC_IEC_60559_DFP__~~
~~#pragma STDC FENV_DEC_ROUND direction~~
~~#endif~~
```

~~Description~~

The ~~FENV_DEC_ROUND~~ pragma is a decimal floating-point analog of the FENV_ROUND pragma. ...

to

The standard pragma

```
#pragma STDC FENV_DEC_ROUND direction
```

is a decimal floating-point analog of the FENV_ROUND pragma. The pragma is provided if and only if the implementation defines __STDC_IEC_60559_DFP__. ...

In 7.12.3 change

~~Synapsis~~

```
~~#include <math.h>~~
~~#pragma STDC FP_CONTRACT on-off-switch~~
```

~~Description~~

The ~~FP_CONTRACT~~ pragma can be used to allow (if the state is "on") or disallow (if the state is "off") the implementation to contract expressions (6.5.1). ~~Each~~ pragma can occur ...

to

The standard pragma

```
#pragma STDC FP_CONTRACT on-off-switch
```

can be used to allow (if the state is "on") or disallow (if the state is "off") the implementation to contract expressions (6.5.1). Use of the pragma does not require inclusion of <math.h>. The pragma can occur ...