Doc. no.: P1233R0
Date: 2018-10-02
Reply-to: Ashley Hedberg (ahedberg@google.com), Matt Calabrese (metaprogrammingtheworld@gmail.com)
Audience: LEWG –> LWG

Shift-by-negative in `shift_left` and `shift_right`

`P0769R2` was applied to the C++ working paper in Rapperswil. That paper defines shifting a range by a negative `n` as a no-op in item (7) of the design decisions section. The LEWG discussion notes from Albuquerque suggest that this design point was not discussed.

Concerns about current behavior

The current treatment of a negative shift as a shift of 0 seems unlikely to match user intent and may hide bugs. If the programmer explicitly wrote a negative value, they probably didn't expect a shift of 0. If the user specified a negative shift as the result of some programmatic calculation, it is likely that the calculation was incorrect, or that a shift in the opposite direction would be the correct behavior. Either way, implicitly shifting by 0 feels questionable.

Proposal

We propose that shifting a range by a negative `n` be a precondition violation; that is, `shift_left` and `shift_right` should require that `n` be greater than or equal to 0. This is consistent with `expr.shift`, which has a precondition that the right operand to `<<` and `>>` must be greater than or equal to 0. Compilers, static analyzers, and other analysis tools could more effectively warn programmers about such shifts if shifting by negative counts was a precondition violation.

Non-Proposals

Reverse shift when shifting by a negative `n`

Some users may expect a shift in the opposite direction when passing a negative `n` to `shift_left` and `shift_right`. The LWG discussion notes on P0769R2 suggest that there are APIs which do this; one example is perlop. This could have a non-trivial cost and is inconsistent with `expr.shift`, so we do not propose it here.

Changing behavior of shifting by large `n`

`expr.shift` has another precondition that the right operand must be less than the length in bits of the left operand. We do not propose changing `shift_left` and `shift_right` to have a similar precondition, as we believe it would be valuable to allow shifting all elements out of a range.

Suggested poll

Do we want the `shift_left` and `shift_right` algorithms to have a precondition that the value of `n` must be greater than or equal to 0?

Acknowledgements

- Dan Raviv and Casey Carter for feedback on an earlier draft of this proposal.