Document number: P1914R0
Revises:
Date: 2019-10-07
Project: ISO JTC1/SC22/WG21: Programming Language C++
Audience: LEWG, LWG
Reply to: Vincent Reverdy
University of Illinois at Urbana-Champaign
vince.rev@gmail.com

On the names of shift algorithms

Abstract

The current draft of the standard introduces `shift_left` and `shift_right` algorithms for C++20. We suggest that the notions of left and right can be misleading in some contexts, including bit containers and strings. We suggest alternative names to remove the ambiguity.

Contents

1 History

- **P1914R0**: original version submitted to LEWG/LWG for the Belfast 2019 Standards Committee Meeting

2 Context

The paper P0769: Add shift to `<algorithm>` introduced two new algorithms to shift sequence of elements, namely `std::shift_left` and `std::shift_right`. Subsequently, the paper P1233: Shift-by-negative in `shift_left` and `shift_right` proposed a new specification for the behaviour of the original shift algorithms when provided with negative inputs. The algorithms and the subsequent modifications have been voted in the standard and are now part of the standard draft N4830 for C++20.

Even if the original version of the proposal P0769R0 included open questions about the naming of the algorithms, the LEWG notes on the 2017 Albuquerque wiki do not report discussions or votes on the chosen names. The open issue section of the original proposal is reported below:

> It would be preferable for `shift_left` and `shift_right` to have more generic names; the fact that the first element in a range is the left-most is a matter of convention which is not specified in the standard, and some programmers may think of the first element as the right most, or maybe the top-most, etc.
> However, `shift_backward`, `shift_back` and `shift_forward` are probably all out of the question, since other algorithms exist, e.g., `std::move_backward` and `std::copy_backward`, in which "backward" means performing the operation starting from the back of the range, instead of from its front.
> `shift_to_front` and `shift_to_back` come to mind; more ideas would be welcome.

We strongly feel that the naming should be discussed, especially in the light of two real-word scenarios in which the chosen names can be very misleading.

3 The problem

3.1 Introduction

As pointed out by the first version of P0769, the notions of left and right are a purely conventional. In practice there is at least two scenarios where the conventions are reversed when compared to what the algorithms are actually doing, leading to ambiguous situations for users:

- bit sequences
- strings for languages that are written from the right to the left

3.2 Bit sequences

Bit iterators as proposed by P0237 are expected to provide a generic way to manipulate bit sequences in a future revision of the standard. To exploit bit parallelism, standard library vendors may provide specializations of the standard algorithms for bits. Experimental implementations of such specializations have been written over the last few years by the author of this proposal and his team at the University of Illinois at Urbana-Champaign. When implementing `std::shift_left` and `std::shift_right` it appeared that to make it work properly with bit sequences, `std::shift_left` had to call the bitwise right shift operator >>, while `std::shift_right` had to call the bitwise left shift operator <<. In practice this quickly escalated into a nightmare of ambiguity, and everything had to be renamed internally to avoid the confusion of library implementers.

3.3 Strings

One can also easily imagine that for languages that are written from the right to the left, the exact same problem will arise. What `std::shift_left` really means is shifting elements towards the beginning of the provided range, and `std::shift_right` to the end of the provided range. In the case of languages that are written from the right to the left, the right corresponds to the beginning of the text while the left corresponds to the end of the text.

3.4 Conclusion

These two cases constitute real-world scenarios in which the convention of left and right is already problematic. The same problem is likely to arise in plenty of more specialized application domains. Since the standard algorithms are meant to be generic, their names should reflect this genericity, as written in the original version of P0769.

4 Proposed resolution

4.1 Option 1: provide one shift algorithm

The first option would be to have only one shift algorithm `std::shift` where the direction would be deduced from the sign of the integer representing the shift amount. A negative integer would translate into a shift towards the beginning of the sequence while a positive integer would translate into a shift towards the end of the sequence. However, this may translate into a branching overhead that may be problematic in low latency and high performance applications.

4.2 Option 2: rename the shift algorithms

The other solutions is simply to rename the existing algorithms with more generic names that removes the ambiguity. Also, we propose the following alternative names:

- `shift_prev/shift_next`
- `shift_begin/shift_end`
- `rshift/shift`
- `shift_to_back/shift_to_front`
- `shift_back/shift_front`
- `shift_backward/shift_forward`
- `shift_backward/shift`
- `backward_shift/forward_shift`
- `backward_shift/shift`
- `back_shift/front_shift`
- `back_shift/shift`
- `backshift/foreshift`
- `backshift/shift`

We strongly encourage readers of this proposal to suggest new names.

5 Acknowledgements

This work has been made possible thanks to the National Science Foundation through the awards CCF-1647432 and SI2-SSE-1642411.

6 References

- P0769R0: Add shift to `<algorithm>`, Dan Raviv, *ISO / IEC – JTC1 / SC22 / WG21* (October 2017)
- P0769R1: Add shift to `<algorithm>`, Dan Raviv, *ISO / IEC – JTC1 / SC22 / WG21* (February 2018)
- P0769R2: Add shift to `<algorithm>`, Dan Raviv, *ISO / IEC – JTC1 / SC22 / WG21* (June 2018)
- P1233R0: Shift-by-negative in `shift_left` and `shift_right`, Ashley Hedberg and Matt Calabrese, *ISO / IEC – JTC1 / SC22 / WG21* (February 2018)
- P1233R1: Shift-by-negative in `shift_left` and `shift_right`, Ashley Hedberg, Matt Calabrese, and Bryce Adelstein Lelbach , *ISO / IEC – JTC1 / SC22 / WG21* (November 2018)
- N4830: Committee Draft, Standard for Programming Language C++, Richard Smith, *ISO / IEC – JTC1 / SC22 / WG21* (August 2019)
- P0237R0: On the standardization of fundamental bitmanipulation utilities, Vincent Reverdy and Robert J. Brunner, *ISO / IEC – JTC1 / SC22 / WG21* (February 2016)
- P0237R10: Wording for fundamental bit manipulation utilities, Vincent Reverdy and Robert J. Brunner, *ISO / IEC – JTC1 / SC22 / WG21* (June 2019)