Document number: D????R0
Date: 2017-07-29
Author: Dan Raviv <dan.raviv@gmail.com>
Audience: LEWG => LWG

Add shift to `<algorithm>`

I. Introduction

This paper proposes adding a shift algorithm to the C++ STL which shifts elements forward or backward in a range of elements.

II. Motivation and Scope

Shifting elements forward or backward in a container or range is a basic operation which the STL should allow performing easily. A main use case is time series analysis algorithms used in scientific and financial applications.

The scope of the proposal is adding a `std::shift` function template to `<algorithm>`. A sample implementation which uses the existing `std::move` and `std::move_backward` can be found here, though it's possible more efficient implementations could be made, since elements are guaranteed to be moved within the same range, not between two different ranges.

III. Expected Objections (and Responses)

1) O: Shifting can be done by using `std::move` (in `<algorithm>`).

R: Which of `std::move` or `std::move_backwards` must[1] be used depends on the shift direction, which is error-prone. It also makes for less readable code; consider

```
std::shift(v.begin(), v.end(), 3);
```

vs.

```
std::move_backward(v.begin(), v.end() - 3, v.end());
```

In addition, std::shift maybe be implemented more efficiently than std::move, since elements are guaranteed to be moved within the same range, not between two different ranges.

---

[1] Technically, in some cases std::move can be used with reverse iterators to implement a shift of elements forward, but not in all cases and it is also less efficient, see https://groups.google.com/a/isocpp.org/d/msg/std-proposals/I76om68B3t0/25bGu7O6AwAJ and https://groups.google.com/a/isocpp.org/d/msg/std-proposals/I76om68B3t0/SPTeZdofAQAJ.

2) O: Instead of shifting a range, you can use a circular buffer.

R: A circular buffer is a valid alternative but one which shouldn't be forced on the programmer, since it does have its own drawbacks, mainly of added use complexity, especially in the case of multiple indices into the buffer, or a mask applied to the buffer which should not cycle with the data. Also, a programmer might need to shift elements in non-circular buffers provided by a 3rd-party library.

3) O: There's already `std::rotate` which is similar in functionality.

R: Shifting just the desired elements would allow for both a more efficient implementation and clearer semantics in case rotation is not needed.

IV. Impact On the Standard

The only impact on the standard is adding the proposed `std::shift` function template overloads to `<algorithm>`.

V. Design Decisions

TODO

VI. Proposed Wording

TODO

VII. Acknowledgements

Thanks to Alexander Zaitsev, Arthur O'Dwyer, Howard Hinnant, Nicol Bolas and Ray Hamel on the std-proposals newsgroup for their helpful comments.