Rangified version of lexicographical_compare_three_way

Document #:	P2022R0
Date:	2020-01-11
Project:	Programming Language C++
Audience:	LEWG LWG

Reply-to:	Ran Regev <regev.ran@gmail.com>

1 Motivation and Scope

This document adds the wording for `ranges::lexicographical_compare_three_way` that is missing in [P1243R2]

2 Proposed Wording

Add to **[algorithm.syn]**

```
template<
  input_iterator I1, sentinel_for<I1> S1,
  input_iterator I2, sentinel_for<I2> S2,
  class Proj1 = identity,
  class Proj2 = identity,
  indirect_binary_predicate<projected<I1,Proj1>, projected<I2,Proj2>> Pred = ranges::less
>
constexpr auto
  ranges::lexicographical_compare_three_way(
    I1 i1, S1, I2 i2, S2, Pred pred = {}, Proj1 = {}, Proj2 = {}
  ) -> std::common_comparison_category_t<decltype(pred(*i1, *i2)), std::strong_ordering>;
```

Add to **§25.7.11 [alg.three.way]**

```
template<class InputIterator1, class InputIterator2, class Cmp>
  constexpr auto
    lexicographical_compare_three_way(InputIterator1 b1, InputIterator1 e1,
                                      InputIterator2 b2, InputIterator2 e2,
                                      Cmp comp)
      -> common_comparison_category_t<decltype(comp(*b1, *b2)), strong_ordering>;
```

Iterators as Input

Option I

```
template<
  input_iterator I1, sentinel_for<I1> S1,
  input_iterator I2, sentinel_for<I2> S2,
  class Proj1 = identity,
  class Proj2 = identity,
  indirect_binary_predicate<projected<I1,Proj1>, projected<I2,Proj2>> Pred = ranges::less
```

```
  >
  constexpr auto
    ranges::lexicographical_compare_three_way(
       I1 i1, S1, I2 i2, S2, Pred pred = {}, Proj1 = {}, Proj2 = {}
    ) -> std::common_comparison_category_t<decltype(pred(*i1, *i2)), std::strong_ordering>;
```

Option II

```
  template<
    input_iterator I1, sentinel_for<I1> S1,
    input_iterator I2, sentinel_for<I2> S2,
    class Proj1 = identity,
    class Proj2 = identity,
    class Pred = ranges::less
  >
  requires indirectly_comparable<I1, I2, Pred, Proj1, Proj2>
  constexpr auto
  ranges::lexicographical_compare_three_way(
    I1 i1, S1, I2 i2, S2, Pred pred = {}, Proj1 = {}, Proj2 = {}
  ) -> std::common_comparison_category_t<decltype(pred(*i1, *i2)), std::strong_ordering>;
```

Option I vs. **Option II**

```
template<
    input_iterator I1, sentinel_for<I1> S1,
    input_iterator I2, sentinel_for<I2> S2,
    class Proj1 = identity,
    class Proj2 = identity,
-   indirect_binary_predicate<projected<I1,Proj1>, projected<I2,Proj2>> Pred = ranges::less
+   class Pred = ranges::less
>
+   requires indirectly_comparable<I1, I2, Pred, Proj1, Proj2>
constexpr auto
    ranges::lexicographical_compare_three_way(
        I1 i1, S1, I2 i2, S2, Pred pred = {}, Proj1 = {}, Proj2 = {}
    ) -> std::common_comparison_category_t<decltype(pred(*i1, *i2)), std::strong_ordering>;
```

Mandates: decltype(pred(*i1, *i2)) is a comparison category type.
Effects: same as `std::lexicographical_compare_three_way`.

Ranges as Input

```
  template<
    input_range R1, input_range R2,
    class Proj1 = identity,
    class Proj2 = identity
    indirect_binary_predicate<projected<iterator_t<R1>,Proj1>, projected<iterator_t<R2>,Proj2>>
  Pred = ranges::less
  >
  constexpr auto
    ranges::lexicographical_compare_three_way(
      R1 r1, R2 r2, Proj1 = {}, Proj2 = {}, Pred pred = {}
    ) -> std::common_comparison_category_t<decltype(pred( *r1.begin(), *r2.begin())),
  std::strong_ordering>;
```

3 Acknowledgements


```
Dan Raviv <dan.raviv@gmail.com>
Michael Park <mcpark@gmail.com> (for github.com/mpark/wg21)
```


4 References

[P1243R2] Dan Raviv. 2019. Rangify New Algorithms.
https://wg21.link/p1243r2