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

2.1 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,
  class Cat = partial_ordering,
  three_way_comparable_with<
    projected<I1,Proj1>, projected<I2,Proj2>, Cat
  > Comp = std::compare_three_way()
>
constexpr auto
  ranges::lexicographical_compare_three_way(
    I1 first1, S1 last1, I2 first2, S2 last2, Comp comp = {}, Proj1 = {}, Proj2 = {}
  ) -> std::common_comparison_category_t<
        decltype(comp(*first1, *first2)), std::strong_ordering>;

template<
  input_range R1, input_range R2,
  class Proj1 = identity,
  class Proj2 = identity
  class Cat = partial_ordering,
  ‘three_way_comparable_with<
    projected<iterator_t<R1>,Proj1>, projected<iterator_t<R2>,Proj2>, Cat
  > Comp = std::compare_three_way()
>
constexpr auto
  ranges::lexicographical_compare_three_way(
    R1&& r1, R2&& r2, Comp comp = {}, Proj1 = {}, Proj2 = {}
  ) -> std::common_comparison_category_t<
        decltype(comp(*r1.begin(), *r2.begin())), std::strong_ordering>;
```

2.2 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

```
template<
  input_iterator I1, sentinel_for<I1> S1,
  input_iterator I2, sentinel_for<I2> S2,
  class Proj1 = identity,
  class Proj2 = identity,
  class Cat = partial_ordering,
  three_way_comparable_with<
    projected<I1,Proj1>, projected<I2,Proj2>, Cat
  > Comp = std::compare_three_way()
>
constexpr auto
  ranges::lexicographical_compare_three_way(
    I1 first1, S1 last1, I2 first2, S2 last2, Comp comp = {}, Proj1 = {}, Proj2 = {}
  ) -> std::common_comparison_category_t<
        decltype(comp(*first1, *first2)), std::strong_ordering>;
```

1 — *Mandates*: decltype(comp(*first1, *first2)) is a comparison category type.

Ranges as Input

```
template<
  input_range R1, input_range R2,
  class Proj1 = identity,
  class Proj2 = identity
  class Cat = partial_ordering,
  three_way_comparable_with<
    projected<iterator_t<R1>,Proj1>, projected<iterator_t<R2>,Proj2>, Cat
  > Comp = std::compare_three_way()
>
constexpr auto
  ranges::lexicographical_compare_three_way(
    R1&& r1, R2&& r2, Comp comp = {}, Proj1 = {}, Proj2 = {}
  ) -> std::common_comparison_category_t<
        decltype(comp(*r1.begin(), *r2.begin())), std::strong_ordering>;
```

2 — *Mandates*: decltype(comp(*r1.begin(), *r2.begin())) is a comparison category type.

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