Implement ranges::starts_with and ranges::ends_with

Zijun Zhao

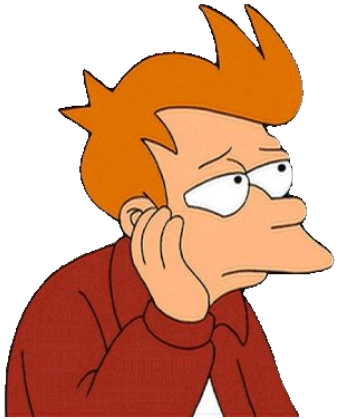
Why we need ranges::starts_with and ranges::ends_with

1. There is no such algorithms for generic input types

std::basic_string<CharT,Traits,Allocator>::starts_with

std::basic_string<CharT,Traits,Allocator>::ends_with

String only : (



Why we need ranges::starts_with and ranges::ends_with

2. Expressive: More readable, simpler and less error-prone

```
const std::vector<Product> prods {
        { "box", 10.0 }, {"tv", 100.0}, {"ball", 30.0},
        { "car", 1000.0 }, {"toy", 40.0}, {"cake", 15.0},
        { "book", 45.0}, {"pc game", 35.0}, {"wine", 25}
};

// the standard version:
std::vector<Product> copy = prods;
std::sort(begin(copy), end(copy), [](const Product& a, const Product& b)
{ return a.name < b.name; }
);

// the ranges version:
std::vector<Product> copy = prods;
std::ranges::sort(copy, {}, &Product::name);
```

Example from https://www.cppstories.com/2022/ranges-alg-part-three/#sort-and-is_sorted

How to implement ranges::starts_with and ranges::ends_with

ranges::starts_with

Input:

```
bool starts_with(I1 first1, S1 last1, I2 first2, S2 last2, Pred pred, Proj1 proj1, Proj2 proj2);
bool starts_with(R1&& r1, R2&& r2, Pred pred, Proj1 proj1, Proj2 proj2);
```

Algorithm:

```
return ranges::mismatch(input).in2 == last2;
```

Output:

true if the second range matches the prefix of the first range, false otherwise.

How to implement ranges::starts_with and ranges::ends_with

ranges::ends_with

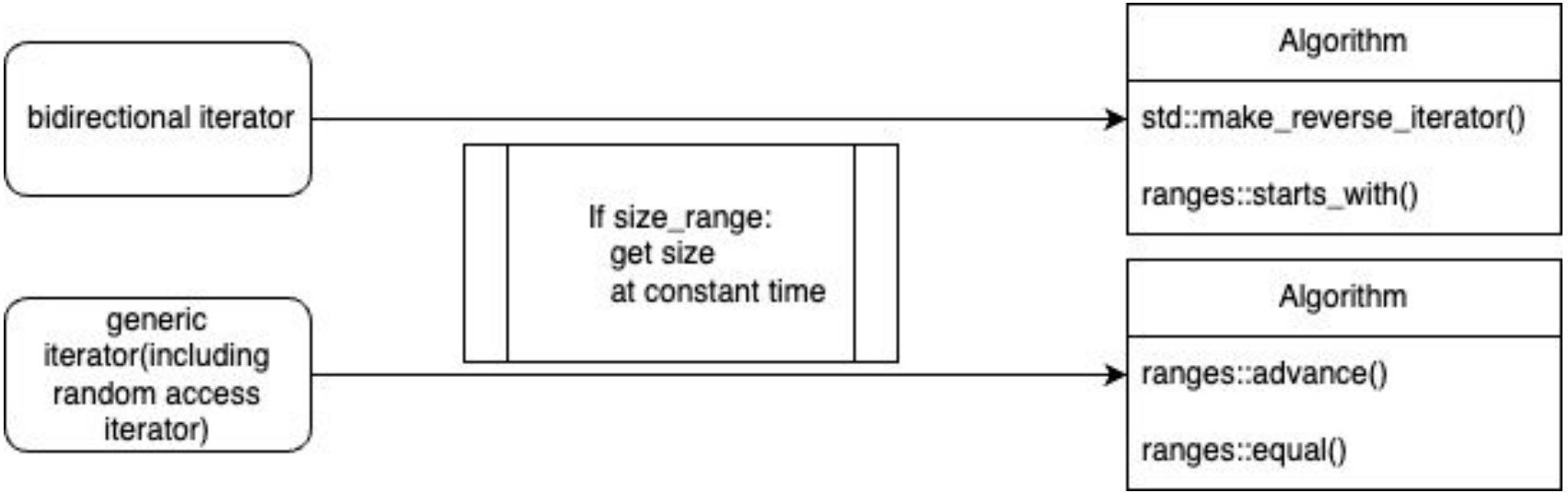
Input:

```
bool ends_with(I1 first1, S1 last1, I2 first2, S2 last2, Pred pred, Proj1 proj1, Proj2 proj2);
bool ends_with(R1&& r1, R2&& r2, Pred pred, Proj1 proj1, Proj2 proj2);
```

Algorithm: See details later

Output:
true if the second range matches the suffix of the first range, false otherwise.

ranges::ends_with algorithm



Performance for different types of iterators

Input type \ size	16	256	4096	65536	1048576	16777216
Bidirectional iterator	7.94ns	126ns	1918 ns	30721ns	494135ns	7889118 ns
Random access iterator	9.77ns	148ns	2254ns	37441ns	583583ns	9178538n s

Input type \ size	16	256	4096	65536	1048576	16777216
Forward iterator	9.06ns	127ns	1925 ns	30991ns	505639ns	8098929 ns
Forward iterator with size optimization	1.10ns	1.10ns	1.10ns	1.09ns	1.10ns	1.10ns

Future work

- Modify these traits to recognize `reference_wrapper` so `ranges::equal()` won't be slower than `ranges::mismatch()`

Thank you for listening to my presentation!