

Parallel STL Algorithms

adjacent_difference, adjacent_find, all_of, any_of, copy, copy_if, copy_n, count, count_if, equal, **exclusive_scan**, fill, fill_n, find, find_end, find_first_of, find_if, find_if_not, **for_each**, **for_each_n**, generate, generate_n, includes, **inclusive_scan**, inner_product, inplace_merge, is_heap, is_heap_until, is_partitioned, is_sorted, is_sorted_until, lexicographical_compare, max_element, merge, min_element, minmax_element, mismatch, move, none_of, nth_element, partial_sort, partial_sort_copy, partition, partition_copy, **reduce**, remove, remove_copy, remove_copy_if, remove_if, replace, replace_copy, replace_copy_if, replace_if, reverse, reverse_copy, rotate, rotate_copy, search, search_n, set_difference, set_intersection, set_symmetric_difference, set_union, sort, stable_partition, stable_sort, swap_ranges, transform, **transform_exclusive_scan**, **transform_inclusive_scan**, **transform_reduce**, uninitialized_copy, uninitialized_copy_n, uninitialized_fill, uninitialized_fill_n, unique, unique_copy

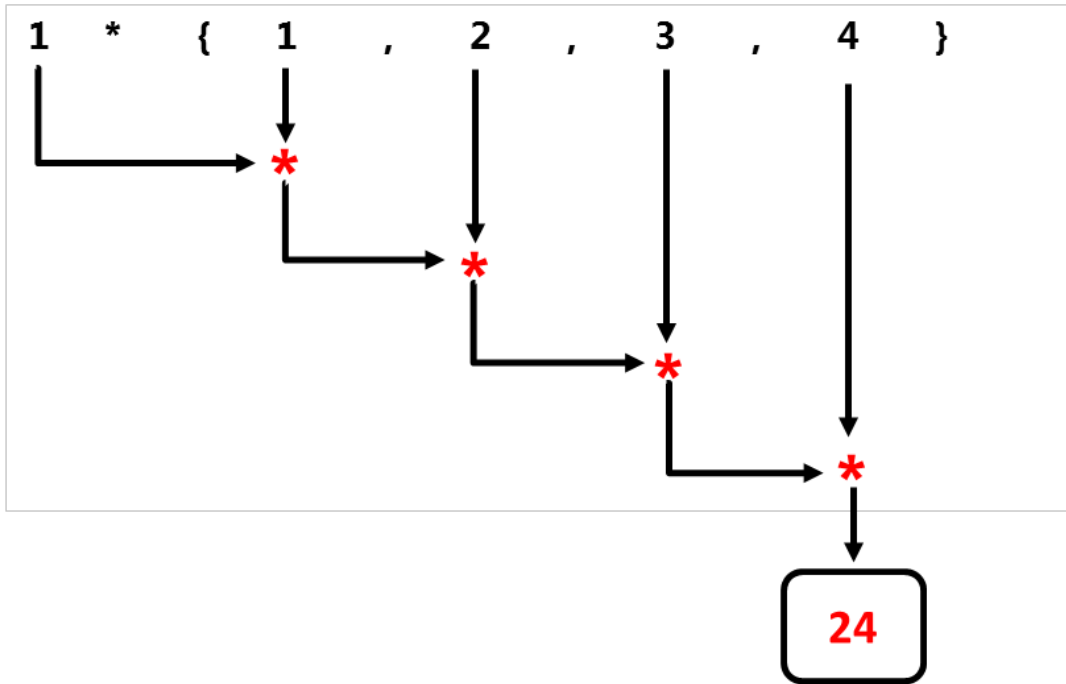
The New C++17 Algorithms

New Algorithm	Description
<code>std::exclusive_scan</code>	Applies from the left a binary callable up to the range's <i>ith</i> (exclusive) element. The left argument of the callable is the previous result. It stores intermediate results.
<code>std::inclusive_scan</code>	Applies from the left a binary callable up to the range's <i>ith</i> (inclusive) element. The left argument of the callable is the previous result. It stores intermediate results.
<code>std::transform_exclusive_scan</code>	Applies first a unary callable to the range and then <code>std::exclusive_scan</code> .
<code>std::transform_inclusive_scan</code>	Applies first a unary callable to the range and then <code>std::inclusive_scan</code> .
<code>std::reduce</code>	Applies a binary callable to the range.
<code>std::transform_reduce</code>	Applies first a unary callable to one or a binary callable to two ranges and then <code>std::reduce</code> to the resulting range.

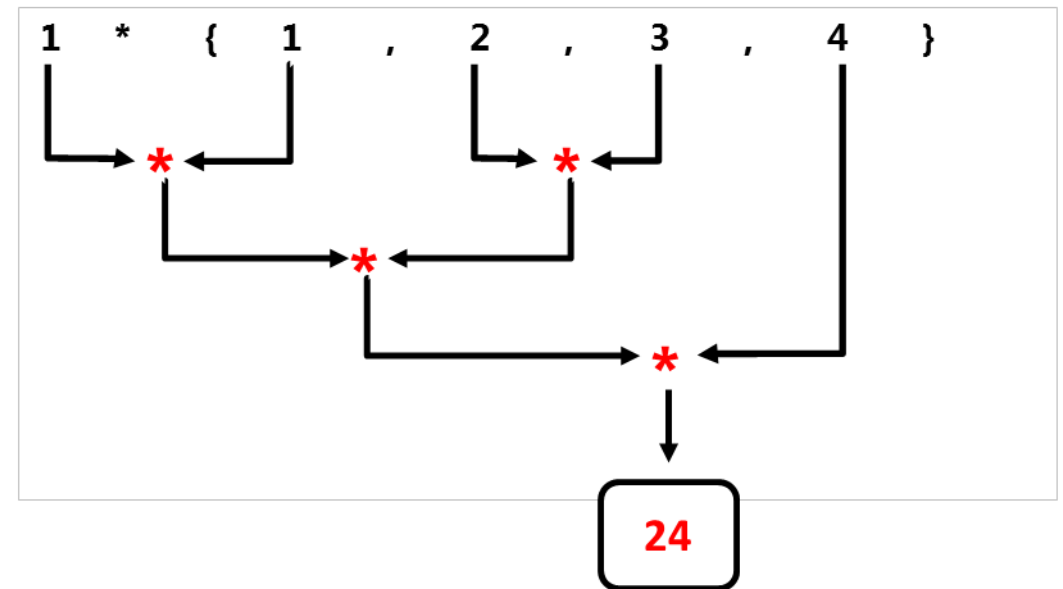
The new C++17 algorithms are designed for parallel execution and need the header `<numeric>`.

std::accumulate versus std::reduce

std::accumulate



std::reduce



std::transform_reduce

std::transform_reduce

- Haskell's function `map` is called `std::transform` in C++
- `std::transform_reduce` ➡ `std::map_reduce`

```
std::vector<std::string> strVec{"Only", "for", "testing", "purpose"};

std::size_t res = std::transform_reduce(std::execution::par,
                                       strVec.begin(), strVec.end(), 0,
                                       [](std::size_t a, std::size_t b){ return a + b; },
                                       [](std::string s){ return s.length(); });

std::cout << res;    // 21
```