Reduce

We often need to reduce a numeric range. That is where std::reduce comes in handy.

WE'LL COVER THE FOLLOWING ^

The six new algorithms that are typically used for parallel execution are also known under the name prefix sum. If the given binary callables are not associative and commutative, the behavior of the algorithms is undefined.

`reduce`:

reduces the elements of the range. `init` is the start value.

- Behaves the same as `std::accumulate` but the range may be rearranged.

```
ValType reduce(InpIt first, InpIt last)
ValType reduce(ExePol pol, InpIt first, InpIt last)

T reduce(InpIt first, InpIt last, T init)
T reduce(ExePol pol, InpIt first, InpIt last, T init)

T reduce(InpIt first, InpIt last, T init, BiFun fun)
T reduce(ExePol pol, InpIt first, InpIt last, T init, BiFun fun)
```

`transform_reduce`:

transforms and reduces the elements of one or two ranges. `init` is the start value.

- Behaves similarly to `std::inner_product` but the range may be rearranged.
- If applied to two ranges
  - if not provided, multiplication is used for transforming the ranges

into one range and addition is used to reduce the intermediate range into the result
    - if provided, `fun1` is used for the transforming step and `fun2` is used for the reducing step
- If applied to a single range
    - `fun2` is used for transforming the given range

```
T transform_reduce(InpIt first, InpIt last, InpIt first2, T init)
T transform_reduce(InpIt first, InpIt last, InpIt first2, T init, BiFun fun1, BiFun fun2)

T transform_reduce(FwdIt first, FwdIt last, FwdIt first2, T init)
T transform_reduce(ExePol pol, FwdIt first, FwdIt last, FwdIt first2, T init, BiFun fun1, BiF

T transform_reduce(InpIt first, InpIt last, T init, BiFun fun1, UnFun fun2)
T transform_reduce(ExePol pol, FwdIt first, FwdIt last, T init, BiFun fun1, UnFun fun2)
```

> **i MapReduce in C++17**
> The Haskell function `map` is called std::transform in C++. When you substitute `transform` with `map` in the name std::transform_reduce, you will get `std::map_reduce`. MapReduce is the well-known parallel framework that first maps each value to a new value, then reduces in the second phase all values to the result.
>
> The algorithm is directly applicable in C++17. In the following example, in the map phase, each word is mapped to its length, and the lengths of all words are then reduced to their sum during the reduce phase. The result is the sum of the length of all words.
>
> ```
> std::vector<std::string> str{"Only", "for", "testing", "purpose"};
>
> std::size_t result = std::transform_reduce(
>                         std::execution::par, str.begin(), str.end
> (), 0,
>                         [](std::size_t a, std::size_t b){ return a
> + b; },
>                         [](std::string s){ return s.length(); }
>                     );
>
> std::cout << result << std::endl;      //   21
> ```