CPU Algorithm Design

Exercise 3 **Students:** Vishal Mangukiya, Konstantin Benz

3.1 Adapting reduce and transform

The input containers in `reduce_LoopUnrolling_view.hpp` and `transform_LoopUnrolling_view.hpp` were adapted to use range-based views as requested in the assignment. In the reduction routines, the memory-backed containers were replaced with `std::views::repeat(1.0f, N)`, using `decltype(std::views::repeat(...))` to define a compatible member variable. This makes the code compute-bound and avoids unnecessary memory usage.
For the transform routines, the input values were changed to `std::ranges::views::iota(0, N)`. Additionally, the output container `W` was replaced by a fixed-size `std::vector<Real>(256)` with modulo-indexed access to enable reuse of the output buffer and simulate non-memory-bound processing.

Challenges encountered:

- Initially, the range variables `V` and `W` were only declared inside each function. However, since multiple functions need access to them, we had to promote them to class-level member variables.
- Using `std::views::repeat` or `std::views::iota` as class members required careful type declarations. Simple type aliases like `std::ranges::repeat_view` or `std::ranges::iota_view` caused type mismatch errors when assigning views with bounds.
- The correct approach was to use `decltype(std::views::repeat(...))` and `decltype(std::views::iota(...))` for the member declarations, as this ensured compatibility with the generated view types and compiler support on the cluster (GCC 14).
- Some loops were not vectorized according to compiler warnings. Since manual unrolling was explicitly requested in the assignment, we did not attempt further refactoring in these cases.

All modified functions were compiled and tested successfully via the targets `reduceVbenchmarkUnroll` and `transformVbenchmarkUnroll`.

3.2 Adapting benchTransformUnrollLoopPeelingDirective

3.3 Adapting benchReduceUnrollTreeDirective

3.4 Adapting benchReduceUnrollSimdXHorizontal and benchReduceUnrollSimdXVertical

3.5 Benchmarking