Lab 7 – Iterators

Exercise 1. Range

Implement class **Range** that defines range for range based loop.

Range(start, stop, step) defines range [start, start+step, start+2*step, …, stop).

It should provide begin and end methods and iterator that goes from start to stop with given step.

By default step=1 and start = 0.

```
for( auto x : Range<int>(1, 9, 2) )
  cout << x <<" ";  // 1 3 5 7
for( auto x : Range<double>(1, 5))
 cout << x << ", ";  // 1, 2, 3, 4,
for( auto x : Range<int>(7) )
  cout << x << " ";  // 0 1 2 3 4 5 6
for( auto x : make_range(1.0, 9.0, 1.5) )
  cout << x << " ";  //  1 2.5 4 5.5 7 8.5
```

Exercise 2. Matrix modifiers

Change implementation of class template Matrix in file **Matrix.h (especially by adding modifiers: const, constexpr, nothrow)**, so that file **ex_7_2_MatrixModifies.cpp** compiles and gives expected output.

Exercise 2. Matrix iterators

Inside a class template Matrix<T,N,M> implement types:

- **iterator** that iterates through the whole matrix in row by row order,
- **const_iterator** that iterates through the whole matrix in row by row order giving read-only access,
- **row_iterator** that iterates through a given row,
- **col_iterator** that iterates through a given column.

All of them should be at least forward iterators (implementing ++, *, →, ==, !=).

Methods

- begin() and end() return an iterator or const_iterator pointing to the first and the past-the-end element in a matrix,
- row_begin(n) and row_end(n) return an iterator pointing to the first and the past-the-end element in the n-th row,
- col_begin(n) and col_end(n) return an iterator pointing to the first and the past-the-end element in the n-th column.

The file **ex_7_3_MatrixIterator.cpp** should compile and work as expected.

Exercise 4. Filter - Lazy evaluation.

Implement function template

make_filter(Container c, Predicate p)

where

- Container c is any class that provides a forward iterator and functions begin() and end().
- Predicate p is a functional object that as its argument takes an element of c and returns bool value.

Function should return an object that will provide a forward iterator, that iterates through all elements in container c for which predicate p returns true. Implement also begin() and end() methods that will return suitable range.

The object should provide lazy evaluation: it does not make a copy of elements that satisfy predicate, it searches for them only if they are needed.

The file **ex_7_4_Filter.cpp** contains use cases and expected output.

Hint: First implement class template Filter<Container, Predicate> that will provide iterator etc. Then function make_filter should just create instance of that class with automatically deduced template parameters.

Hint: If argument c is an lvalue than object should store reference to it, for rvalues make a copy to avoid dangling references. Use perfect forwarding to achieve that.