Parallel STL

- Execution policies
  - `std::execution::seq`
    - sequential in one thread
  - `std::execution::par`
    - parallel
  - `std::execution::par_unseq`
    - parallel and vectorized (SIMD)
    - interleaving of individual loops allowed
  - `std::execution::unseq` (C++20)
    - parallel and vectorized (SIMD)
    - interleaving of individual loops is not allowed

Parallel STL

```
const int SIZE = 8;
int vec[]={1, 2 , 3, 4, 5, 6, 7, 8};
int res[SIZE] = {0,};

int main(){
    for (int i= 0; i < SIZE; ++i){
        res[i] = vec[i] + 5;
    }
}
```

Not vectorized

```
movslq   -8(%rbp), %rax
movl     vec(,%rax,4), %ecx
addl     $5, %ecx
movslq   -8(%rbp), %rax
movl     %ecx, res(,%rax,4)
```

Vectorized

```
movdqa  .LCPI0_0(%rip), %xmm0    # xmm0 = [5,5,5,5]
movdqa  vec(%rip), %xmm1
paddd   %xmm0, %xmm1
movdqa  %xmm1, res(%rip)
paddd   vec+16(%rip), %xmm0
movdqa  %xmm0, res+16(%rip)
xorl    %eax, %eax
```

Parallel STL

```
std::vector<int> vec = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};


std::sort(vec.begin(), vec.end());                              // sequential as ever

std::sort(std::execution::seq, vec.begin(), vec.end());         // sequential

std::sort(std::execution::par, vec.begin(), vec.end());         // parallel

std::sort(std::execution::par_unseq, vec.begin(), vec.end()); // par + vec
                                                                // loops can interleave

std::sort(std::execution::unseq, vec.begin(), vec.end());       // par + vec
                                                                // loops cannot interleave
```

exceptionExecutionPolicy.cpp