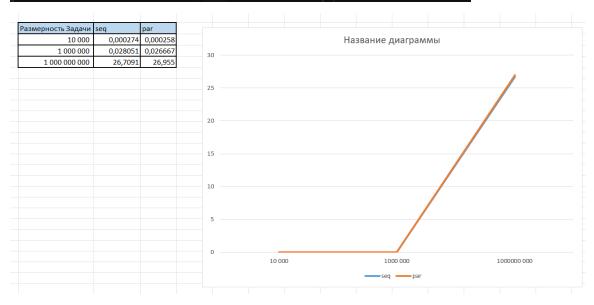
## Доскоч Роман 13 группа 3 курс ТП Домашнее задание от 12.11.2021

## Исследовать функцию std::generate

```
template< class ExecutionPolicy, class ForwardIt, class Generator >
void generate ( ExecutionPolicy&& policy, ForwardIt first, ForwardIt last,
Generator g );
Код
#include <execution>
#include <algorithm>
#include <iostream>
#include <vector>
#include <chrono>
#include <concepts>
enum class Policy { seq, par, par_unseq };
// Функция выбора метода выполнения : последовательно или параллельно.
template<class F>
auto maybe_parallel(F f, const Policy p) {
         switch (p) {
         case Policy::seq: return f(std::execution::seq);
         case Policy::par: return f(std::execution::par);
         default: return f(std::execution::par_unseq);
// Обертка функции std::generate() с параметром Policy.
auto policy_generate(std::vector<int>& vec, Policy p) {
         return maybe_parallel(
                 [&](auto& pol)
                          return std::generate(pol, vec.begin(), vec.end(),
                                   [x = 2]()mutable
                                            return x = pow(cos(x) * sin(x),2);
                                   });
                  }, p);
// Функция тестирования.
auto generate_test(const Policy policy, int n) {
         std::vector<int>v(n);
         auto start = std::chrono::high resolution clock::now();
         policy_generate(v, policy);
        auto stop = std::chrono::high_resolution_clock::now();
         return std::chrono::duration_cast<std::chrono::microseconds>(stop - start).count() / 1e6;
}
int main() {
         setlocale(LC ALL, "ru");
         for (auto n: { 10000, 1000000, 1000000000 }) {
                  auto time = generate_test(Policy::seq, n);
                  auto par_time = generate_test(Policy::par, n);
                  std::cout << "Размерность " << n << ": \n"
                          << "Линейный метод: " << time << std::endl
                          << "Параллельный метод: " << par_time << std::endl
                          << "Параллельный метод был быстрее в: "
                          << (time - par time) / par time << "pa3.\n\n";
         }
```



Вывод: видно, что использование параллельной реализации метода generate, мало чем отличается от последовательного подхода. Объясняется это тем, что алгоритм генерации сложен для распараллеливания. Время так же зависит от функции, на которой будет генерироваться массив. Ее сложность прямой показатель времени выполнения.