

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский химико-технологический университет имени Д.И.
Менделеева»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

Выполнил студент группы КС-36: Золотухин А.А.

Ссылка на репозиторий: [https://github.com/
CorgiPuppy/
parallel-prog-labs](https://github.com/CorgiPuppy/parallel-prog-labs)

Принял: Бабкин Михаил Андреевич

Дата сдачи: 01.10.2025

Москва
2025

Оглавление

Описание задачи	1
Выполнение задачи	1

Описание задачи

Выполнение задачи

```
1 #include <iostream>
2 #include <vector>
3 #include <random>
4 #include <algorithm>
5 #include <chrono>
6 #include <fstream>
7 #include <thread>
8
9 #include "../include/Constants.h"
10
11 void generateVector(std::vector<int> &);
12 void calculateProductParallel(std::vector<int>, std::vector<int>);
13 void calculateScalarProduct(const std::vector<int>&, const std::vector<int>&, int,
    int, int&);
14
15 int main() {
16     std::ofstream time_long(Constants::folder + "time-long.dat");
17
18     for (int i = 0; i < (int)Constants::longOfVector.size(); i++) {
19         std::vector<int> vectorA(Constants::longOfVector[i]);
20         std::vector<int> vectorB(Constants::longOfVector[i]);
21
22         generateVector(vectorA);
23         generateVector(vectorB);
24
25         std::chrono::high_resolution_clock::time_point start = std::chrono::
high_resolution_clock::now();
26
27         calculateProductParallel(vectorA, vectorB);
28
29         std::chrono::high_resolution_clock::time_point end = std::chrono::
high_resolution_clock::now();
30         std::chrono::duration<long double, std::milli> milli_diff = end - start;
31         long double time_taken = milli_diff.count();
32
33         time_long << Constants::longOfVector[i] << " " << time_taken << std::endl;
34     }
35
36     time_long.close();
37
38     return 0;
39 }
40
41 void generateVector(std::vector<int> &vector) {
42     std::random_device rnd_device;
43     std::mt19937 mersenne_engine {rnd_device()};
44     std::uniform_int_distribution<int> dist {Constants::minNumberOfVector, Constants
::maxNumberOfVector};
45
46     auto gen = [&]() {
47         return dist(mersenne_engine);
48     };
49
50     std::generate(vector.begin(), vector.end(), gen);
```

```

51 }
52
53 void calculateProductParallel(std::vector<int> firstVector, std::vector<int>
secondVector) {
54     int total_scalar_product = 0;
55
56     std::vector<std::thread> threads;
57     std::vector<int> partial_results(Constants::numberOfThreads, 0);
58
59     for (int i = 0; i < Constants::numberOfThreads; i++) {
60         int start = i * (firstVector.size() / Constants::numberOfThreads);
61
62         int end;
63         if (i == Constants::numberOfThreads - 1)
64             end = firstVector.size();
65         else
66             end = (i + 1) * (firstVector.size() / Constants::numberOfThreads);
67
68         threads.emplace_back(calculateScalarProduct, std::cref(firstVector), std:::
cref(secondVector), start, end, std::ref(partial_results[i]));
69     }
70
71     for (auto& thread : threads) {
72         thread.join();
73     }
74
75     for (int i = 0; i < Constants::numberOfThreads; i++) {
76         total_scalar_product += partial_results[i];
77     }
78 }
79
80 void calculateScalarProduct(const std::vector<int>& firstVector, const std::vector<
int>& secondVector, int start, int end, int& result) {
81     result = 0;
82     for (int i = start; i < end; i++) {
83         result += firstVector[i] * secondVector[i];
84     }
85 }

```

Расчётная кривая зависимости времени выполнения от длины вектора

