**Отчет**

**Код программы:**

#include <iostream>

#include <queue>

#include <tuple>

#include <condition\_variable>

#include <thread>

#include <cstdio>

#include <string>

#include <fstream>

using namespace std;

using namespace chrono\_literals;

#pragma warning(disable:4996)

queue<string> q;

mutex mut;

condition\_variable cv;

bool finished{ false };

ofstream output;

ifstream input;

static void producer() {

input.open(R"(words.txt)");

while (!input.eof()) {

lock\_guard<mutex> lk{ mut };

string s;

input >> s;

q.push(s);

cv.notify\_all();

}

lock\_guard<mutex> lk{ mut };

finished = true;

cv.notify\_all();

input.close();

}

static void consumer() {

output.open(R"(inverted\_words.txt)");

while (!finished) {

unique\_lock<mutex> l{ mut };

cv.wait(l, [] { return !q.empty() || finished; });

while (!q.empty()) {

string s = q.front();

reverse(s.begin(), s.end());

output << s << endl;

q.pop();

}

}

output.close();

}

void parallel() {

thread t1{ producer };

thread t2{ consumer };

t1.join();

t2.join();

}

void sequential() {

input.open(R"(words.txt)");

output.open(R"(inverted\_words.txt)");

while (!input.eof()) {

string s;

input >> s;

reverse(s.begin(), s.end());

output << s << endl;

}

input.close();

output.close();

}

int main() {

auto start\_time = clock();

parallel();

auto end\_time = clock();

double parallel\_time = (double)(end\_time - start\_time);

std::cout << "Parallel time :" << parallel\_time << endl;

start\_time = clock();

sequential();

end\_time = clock();

double sequent\_time = (double)(end\_time - start\_time);

std::cout << "Sequent time :" << sequent\_time << endl;

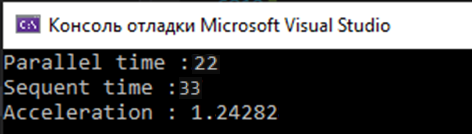
double acceleration\_factor = sequent\_time / parallel\_time;

std::cout << "Acceleration : " << acceleration\_factor << endl;

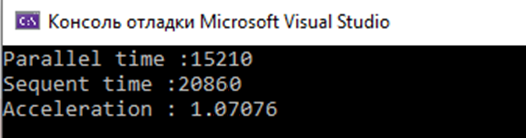
}

**Результат:**

* **Размер 10 000 строк:**

****

* **Размер 1 200 000 строк:**

****