#include <iostream>

#include <vector>

#include <thread>

#include <chrono>

#include <functional>

using namespace std;

int N = 10000000;

int p = 4;

vector<int> a, b, c1(N), c2(N);

void add(const int start, const int step, const int N, function<int(int, int)>f) {

for (int i = start; i < N; i += step)

c1[i] = f(a[i], b[i]);

}

void sequencialAdd(const int start, const int step, const int N, function<int(int, int)>f) {

auto t\_start = chrono::steady\_clock::now();

for(int i = 0; i < N; i ++)

c1[i] = f(a[i], b[i]);

auto t\_finish = chrono::steady\_clock::now();

cout << "Sequencial time: " << chrono::duration <double, milli>(t\_finish - t\_start).count() << "ms\n";

}

int main() {

for (int i = 0; i < N; i++) {

a.push\_back(1);

b.push\_back(1);

//c.push\_back(0);

}

sequencialAdd(0, p, N, [=](int xx, int yy) { return xx + yy; });

auto t\_start = chrono::steady\_clock::now();

vector<thread> threads;

for (int i = 0; i < p; i++) {

threads.push\_back(thread(add, i, p, N, [=](int xx, int yy) { return xx + yy; }));

}

for (int i = 0; i < p; i++) {

threads[i].join();

}

auto t\_finish = chrono::steady\_clock::now();

cout << "Paralel time: " << chrono::duration <double, milli>(t\_finish - t\_start).count() << "ms\n";

return 0;

}