Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОННОЙ ТЕХНИКИ»

Институт системной и программной инженерии

и информационных технологий (СПИНТех)

**Отчёт**

по дисциплине «Технологии параллельного программирования»

**Лабораторная работа №3**

**Вариант-7**

Руководитель

\_\_\_\_\_\_\_\_\_\_ Можжухина A. В.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Студент группы ПИН-23

\_\_\_\_\_\_\_\_\_\_ Исламов Р. Р.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

*Москва*

*2023*

# 1 Листинг

// Lab3.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//Исламов Вариант 7

//

// Задание 1. В соответствии с вариантом задания, написать на C++ программу программу, реализующую многопоточность на основе технологии OpenMP: она должна вычислять значение функции с помощью редукции. Измерять время работы программ для различных значений параметра N. Результаты занести в отчёт.

//

// Вариант 2. На основе двух равно размерных массивов A и B(длины N) функция возвращает сумму значений, больших 1.

//

// Задание 2. Модифицировать программу, составленную по Заданию 1, убрав возможность параллельного использования вычислительных ресурсов. Измерять время работы программы для тех же значений параметров, что были использованы при выполнении Задания 1. Результаты сравнить и занести в отчёт.

#include <omp.h>

#include <iostream>

#include <locale.h>

const int n = 500;

void init\_rand(int\* a) {

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

a[i] = rand() % 10; //запись в матрицу случайных чисел от 1 до 9

//std::cout << std::endl;

}

}

void show\_matrix(int\* a) {

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

std::cout << a[i] << " ";

std::cout << std::endl;

}

}

void OpenMP(int a[n], int b[n]) {

int i;

int sum = 0;

#pragma omp parallel shared(a)

{

#pragma omp for private(i) reduction(+:sum)

for (i = 0; i < n; i++)

{

int value = std::max(a[i] + b[i], 4 \* a[i] - b[i]);

if (value > 1)

sum += value;

}

} /\* Завершение параллельного фрагмента \*/

printf("Сумма элементов матрицы вычесенных по условию равна %i\n", sum);

}

void noOpenMP(int a[n], int b[n]) {

int i;

int sum= 0;

#pragma omp parallel shared(a)

{

#pragma omp for private(i) reduction(+:sum)

for (i = 0; i < n; i++)

{

int value = std::max(a[i] + b[i], 4 \* a[i] - b[i]);

if (value > 1)

sum += value;

}

} /\* Завершение параллельного фрагмента \*/

printf("Сумма элементов матрицы вычесенных по условию равна %i\n", sum);

}

int main()

{

setlocale(LC\_ALL, "Russian");

int a[n], b[n];

init\_rand(a);

init\_rand(b);

//show\_matrix(a);

double noopenMp\_start\_time = omp\_get\_wtime();

noOpenMP(a,b);

double noopenMp\_end\_time = omp\_get\_wtime();

std::cout << noopenMp\_end\_time - noopenMp\_start\_time << std::endl;

double openMp\_start\_time = omp\_get\_wtime();

OpenMP(a,b);

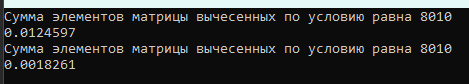
double openMp\_end\_time = omp\_get\_wtime();

std::cout << openMp\_end\_time - openMp\_start\_time << std::endl;

return 0;

}

# 2 Результат работы



N = 500