Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования

**«Национальный исследовательский ядерный университет «МИФИ»**

**ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ**

**Кафедра №42 (криптологии и кибербезопасности)**

ЛАБОРАТОРНАЯ РАБОТА №2  
по дисциплине: «Параллельное программирование»  
на тему: «Выделение ресурса параллелизма. Технология OpenMP»

**Выполнил:**студент группы Б19-515  
Родионов Дмитрий Александрович

**Москва, 2021**

Оглавление

[Используемая рабочая среда 3](#_Toc83734170)

[Анализ приведённого алгоритма 4](#_Toc83734171)

[Заключение 5](#_Toc83734172)

# Используемая рабочая среда

**Процессор:** Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.59 GHz

**Оперативная память:** 16.0 ГБ

**Тип системы:** 64-разрядная операционная система, процессор x64

**Версия:** 21H1

**Среда разработки:** WSL2 Ubuntu 20.04 LTS

**Поддерживаемая версия OpenMP:** 4.5

# Анализ приведённого алгоритма

**Структура алгоритма поиска заданного элемента массива:**

1) параллельный участок кода – в данном блоке программы происходит распараллеливание классического цикла сравнения элементов с целевым значением, т. е. в каждом используемом потоке осуществляется определённое число сравнений различных элементов массива с общей переменной *target* и изменение общей переменной *label*, соответствующее результату очередного произведённого сравнения; множество итераций цикла в рассматриваемом алгоритме разделяется на куски примерно одинакового размера, которые распределяются между используемыми нитями; все сформированные фрагменты имеют одинаковый размер, за исключением последнего куска, в котором единственный поток обрабатывает ранее не задействованные элементы массива.

2) последовательный участок кода – в данном блоке программы происходит непосредственно вычисление среднего времени работы алгоритма, а также вывод в консоль искомого индекса заданного элемента массива.

**Назначение используемых директив OpenMP и их опций:**

1) *parallel* – задаёт параллельную область программы (строки №40-66):

1. *num\_threads(threads)* – явное задание количества нитей, которые будут выполнять параллельную область;
2. *shared(array, count, target, index, threads)* – задаёт список переменных, общих для всех нитей;
3. *default(shared)* – всем переменным в параллельной области, которым явно не назначен класс, будет назначен класс *shared*.

В случае отсутствия данной директивы в приложенном коде алгоритма, всё множество итераций цикла выполнялось бы единственным потоком.

2) *for* – распределение итераций цикла между различными нитями: относится к идущему следом блоку, включающему оператор *for* (строки №43-54).

В случае отсутствия данной директивы в приложенном коде алгоритма, всё множество итераций цикла выполнялось бы каждым порождённым потоком.

3) *single* – выделение параллельного участка кода, который требуется выполнить лишь один раз (строки №58-65).

В случае отсутствия данной директивы в приложенном коде алгоритма, всё множество итераций цикла выполнялось бы каждым порождённым потоком.

# Заключение

В данной лабораторной работе проведён теоретический и экспериментальный анализ параллельного алгоритма, реализованного с использованием технологии параллельных вычислений OpenMP. Для достижения поставленной цели раскрыты понятия временной сложности, ускорения и эффективности программы; изучены базовые способы математического описания алгоритма. Также в ходе выполнения работы были косвенно определены характеристики используемого процессора.

Цель, поставленная в начале работы, достигнута; задачи выполнены.