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 №1  
по дисциплине: «Параллельное программирование»  
на тему: «Введение в параллельные вычисления. Технология OpenMP»

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

**Москва, 2021**

Оглавление

[Используемая рабочая среда 3](#_Toc83062564)

[Анализ приведённого алгоритма 4](#_Toc83062565)

[Заключение 5](#_Toc83062566)

# 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 максимального элемента массива:**

1) параллельный участок кода – в данном блоке программы происходит распараллеливание классического цикла сравнения элементов в оставшейся части массива, т. е. в каждом используемом потоке осуществляется определённое число сравнений различных элементов массива с локальной копией общей переменной *max* и изменение данной локальной копии, соответствующее результату очередного произведённого сравнения; множество итераций цикла в рассматриваемом алгоритме разделяется на куски примерно одинакового размера, которые распределяются между используемыми нитями;

2) последовательный участок кода – в данном блоке программы происходит определение максимального среди всех отработавших нитей значения, т. е. непосредственно нахождение максимального элемента исходного массива, а также его вывод в консоль.

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

1) *parallel* – задаёт параллельную область программы (строки №30-38):

1. *num\_threads(threads)* – явное задание количества нитей, которые будут выполнять параллельную область;
2. *private(iterations, start\_time, end\_time)* – задаёт список переменных, для которых порождается локальная копия в каждой нити;
3. *shared(array, count)* – задаёт список переменных, общих для всех нитей;
4. *reduction(max: max)* – задаёт оператор и список общих переменных; для каждой переменной создаются локальные копии в каждой нити; локальные копии инициализируются соответственно типу оператора; над локальными копиями переменных после выполнения всех операторов параллельной области выполняется заданный оператор;
5. *default(none)* – всем переменным в параллельной области класс должен быть назначен явно.

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 №32-36).

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 параллельных алгоритмов, реализованных 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 использовались средства автоматизации выполнения программ в операционной системе Linux Ubuntu 20.04 LTS.

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