Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное
образовательное учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Высшая школы электроники и компьютерных наук
Кафедра системного программирования

ОТЧЕТ
о лабораторной работе №1
по дисциплине «Технологии параллельного программирования»

Выполнил:
студент группы КЭ-220
_____/Голенищев А. Б.
_____ 2024 г.

Отчет принял:
_____/Жулев А. Э.
_____ 2024 г.

Челябинск
2024

Задание 1. Создание проекта в среде MS Visual Studio с поддержкой OpenMP

IDE: Qt Creator.

Создали проект приложения на C++, система сборки qmake, компилятор MinGW. Открываем файл проекта с расширением *.pro, подключаем поддержку OpenMP в несколько строк:

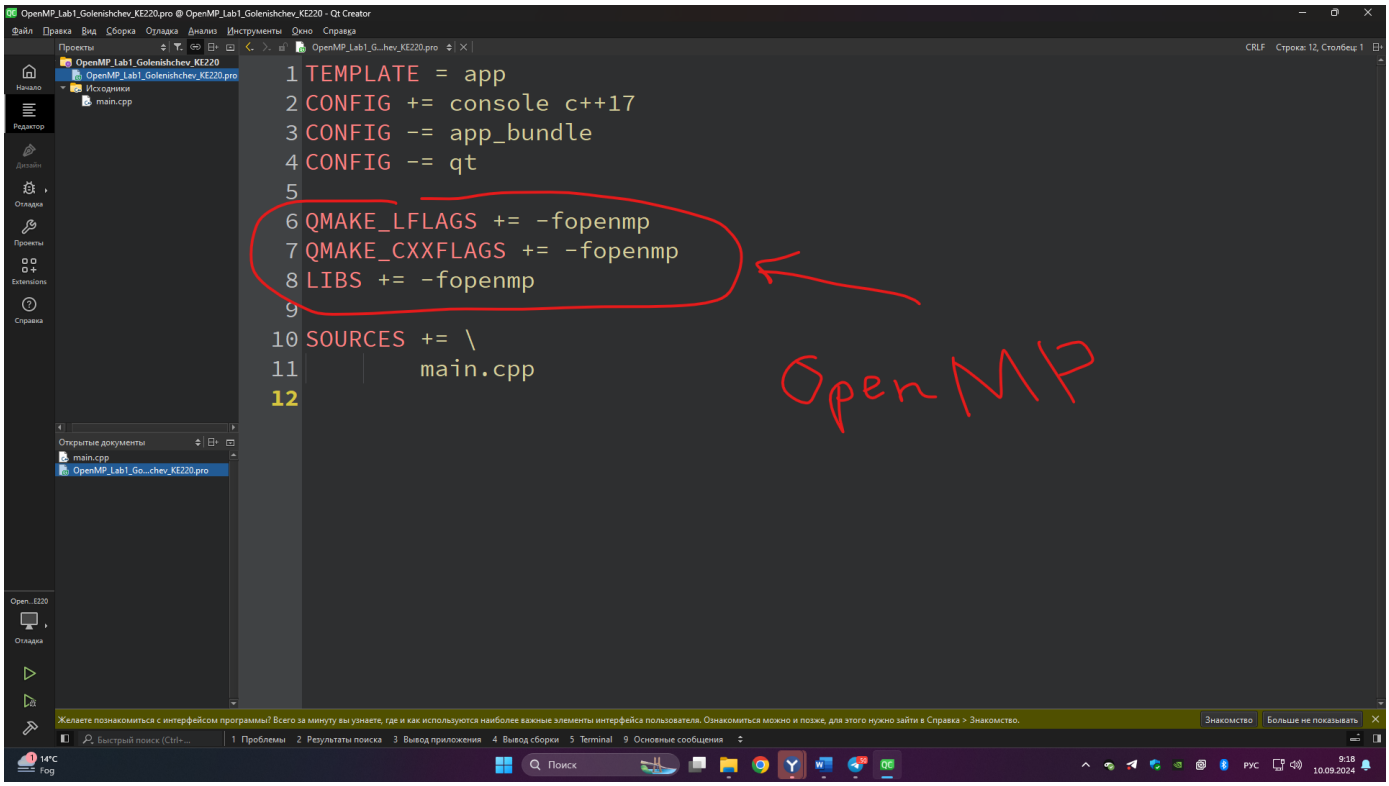

Рисунок 1. Добавление поддержки OpenMP в проект с qmake

Задание 2. Многопоточная программа «Hello World!»

Написали первую многопоточную программу, результат работы представлен на рисунке 2. Процессор: Intel Core i7-13700K (16 ядер, 24 потока). В выводе количество выведенных “Hello World!” соответствует количеству потоков процессора.

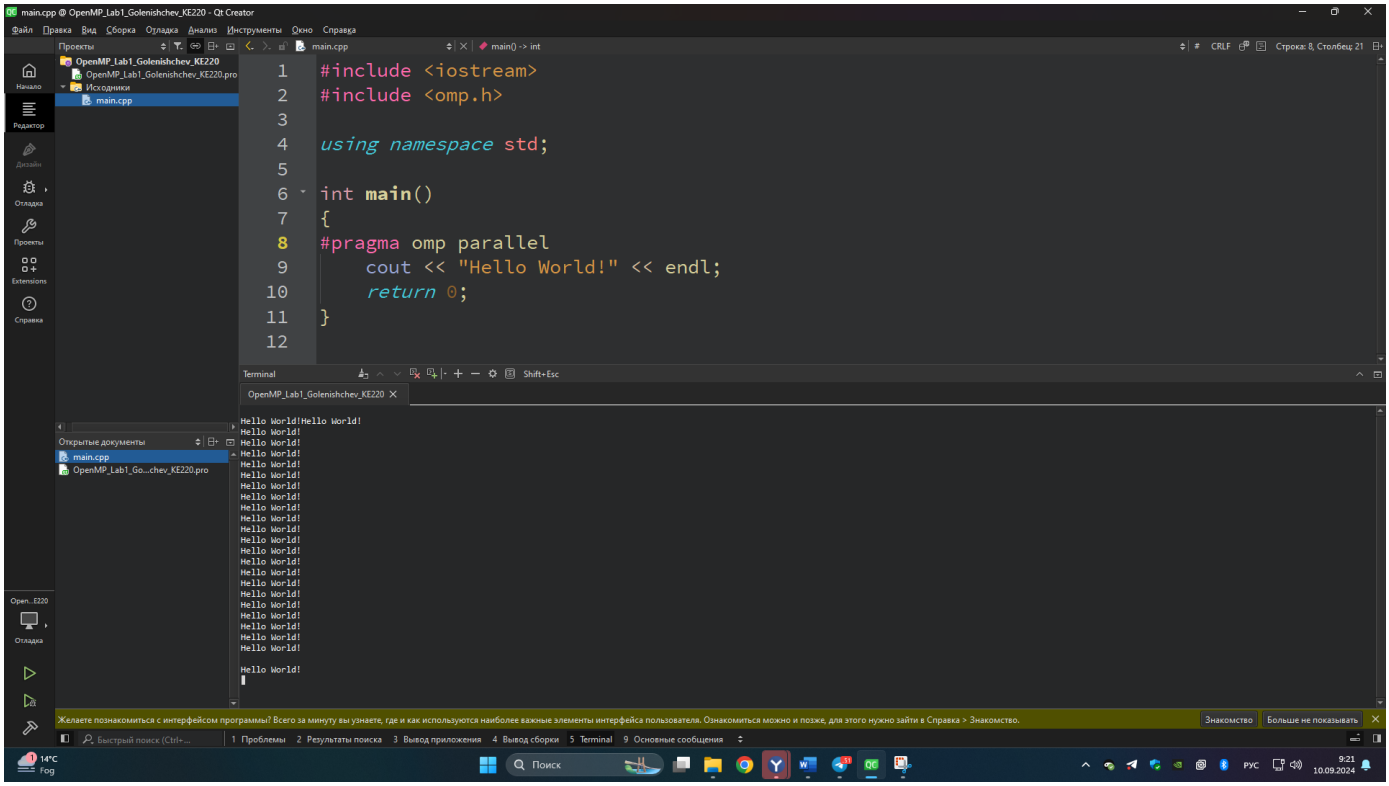

Рисунок 2. Первая многопоточная программа

Задание 3. Программа «I am!»

Написали программу, в которой создается k нитей, и каждая нить выводит на экран свой номер и общее количество нитей в параллельной области в формате: I am <thread> from <threads>! Программа работает правильно, но мы получили паразитный эффект распараллеливания – дублирование «I am» при выводе в консоль.

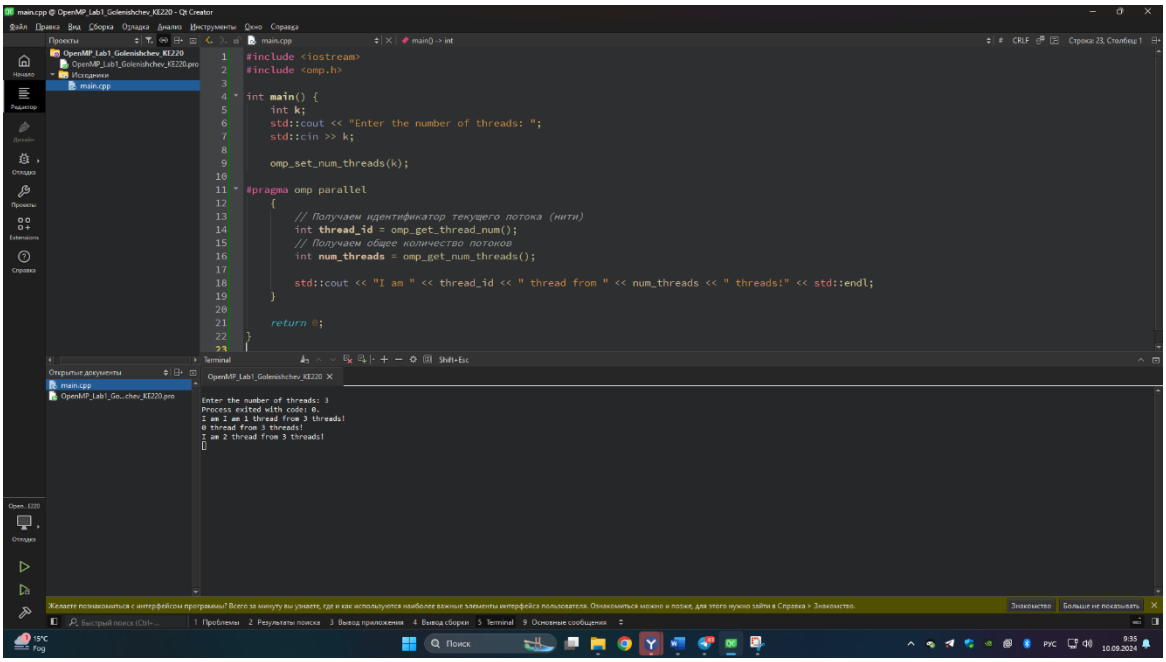

Рисунок 3. Программа, создающая k-нитей

Вывод строк с четным номером, могут строки выводиться в любом порядке:

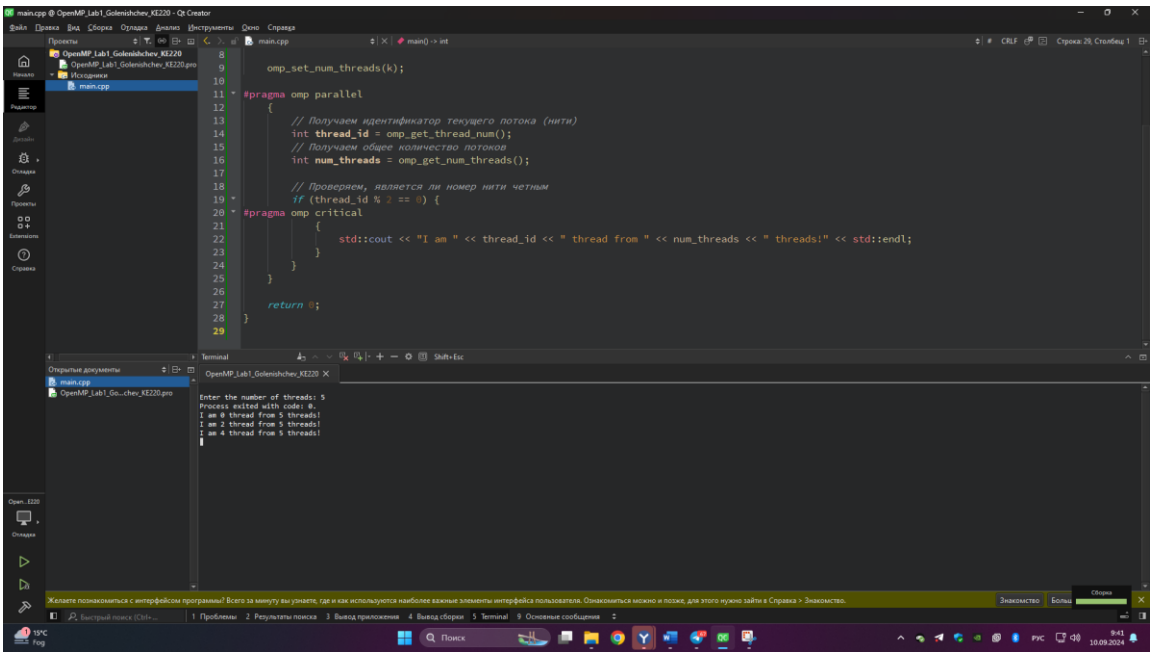

Рисунок 4. Вывод строк с четным номером (ноль – это четное число)

Ответы на вопросы к лабораторной работе:

1. Что такое OpenMP? Какие модели он реализует? Опишите модели и их связь.

   OpenMP – это стандарт интерфейса для многопоточного программирования над общей памятью и набор средств компилируемых языков программирования C++ и Fortran. Он предоставляет набор директив, библиотек и переменных окружения.

   Модели:

   SPMD-модель (Single Program Multiple Data) – одна программа, выполняемая на многих процессорах.

2. В каких языках реализован этот стандарт? Из каких частей состоит реализация в Visual Studio? (вместо VS отвечу про Qt)

   C++ и Fortran. Qt – это фреймворк, включающий в себя набор инструментов для программирования на C++, Python. В состав SDK с версии Qt 4.6 включена поддержка OpenMP (компилятор MinGW с ее поддержкой), пример с использованием qmake был приведен в задании 1. Qt Creator – свободная IDE, является отдельным продуктом Qt Company.

3. Какие существуют варианты задания количества нитей в параллельном регионе? Сколько нитей будет создано, если указаны оба варианта с разными значениями? Что конкретно делает функция omp_set_num_threads()?

omp_set_num_threads(int num_threads): Устанавливает глобальное количество нитей для всех параллельных регионов.

Директива #pragma omp parallel num_threads(num_threads): Задаёт количество нитей для конкретного параллельного региона (имеет приоритет).

Переменная окружения OMP_NUM_THREADS: Определяет количество нитей на уровне окружения.

Если указаны оба варианта, директива num_threads имеет приоритет. Функция omp_set_num_threads() устанавливает глобальное количество нитей для всех последующих параллельных регионов.

4. Как идентифицируются нити в OpenMP? Для чего это нужно? Приведите содержательный пример. Совпадают ли эти идентификаторы с идентификаторами потоков в ОС?
    1. ***Нити идентифицируются с помощью* omp_get_thread_num(), *который возвращает номер нити в пределах параллельного региона (начиная с 0).***
    2. ***Это нужно для распределения задач между нитями и синхронизации работы.***
    3. ***Пример: каждая нить обрабатывает свой блок данных.***
    4. ***Идентификаторы OpenMP не совпадают с идентификаторами потоков ОС.***
5. Каков порядок вывода сообщений нитями? Всегда ли он одинаков? Чем определяется этот порядок?
    1. ***Порядок вывода не гарантируется и может меняться при каждом запуске.***
    2. ***Определяется операционной системой и планировщиком потоков.***
    3. ***Для контролируемого порядка вывода используется* #pragma omp critical, *барьеры или атомарные операции.***

Выводы:

Изучили стандарт OpenMP для организации параллельного программирования на языке C++. Рассмотрены методы задания количества нитей в параллельных регионах, особенности идентификации нитей, а также поведение программы при выводе данных из разных потоков. Практическая реализация показала, как можно контролировать выполнение параллельных участков кода и распределение задач между нитями. Также было выявлено, что порядок вывода сообщений из нитей непредсказуем и может зависеть от множества факторов, включая планировщик потоков операционной системы.