ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК

Домашняя работа № 4.

Разработка многопоточных приложений с использованием OpenMP.

Выполнила студент БПИ196 Филиппова Мария

Москва

2020

**Задание.**

**Вариант 24.**

**Задача о Пути Кулака.**

На седых склонах Гималаев стоят два древних буддистских монастыря: Гуань-Инь и Гуань-Янь. Каждый год в день сошествия на землю боддисатвы Араватти монахи обоих монастырей собираются на совместное празднество и показывают свое совершенствование на Пути Кулака. Всех соревнующихся монахов разбивают на пары, победители пар бьются затем между собой и так далее, до финального поединка. Монастырь, монах которого победил в финальном бою, забирает себе на хранение статую боддисатвы. Реализовать многопоточное приложение, определяющее победителя. В качестве входных данных используется массив, в котором хранится количество энергии Ци каждого монаха. При решении использовать принцип дихотомии.

**Использованная модель**: «производители и потребители» с древовидной организацией потоков – в соответствии с требованием об использовании принципа дихотомии. Дихотомия – последовательное деление класса на два взаимоисключающих подкласса. Что касается модели «производители и потребители», то она заключается в следующем: процесс-производитель генерирует информацию в некотором буфере, которая используется процессом -потребителем. При переполнении буфера производитель должен подождать, пока в буфере не появится свободное место; при исчерпании буфера потребитель должен ждать появления в нем нового элемента. Часто производители и потребители в конвейер – последовательность процессов, в которой каждый потребляет данные предшественника и поставляет данные для следующего процесса. В данной программе использована древовидная структура организации процессов. Выглядит она следующим образом:

результат

результат

входные данные

результат

результат

входные данные

результат

результат

входные данные

Конвейер

... ... ...

**Алгоритм решения задачи**.

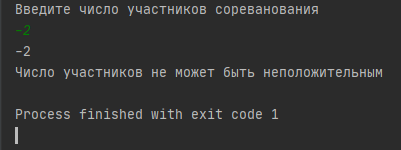
В метод передается массив с информацией об уровнях ци монахов, переменная для хранения наибольшего значения уровня, а также переменная для хранения индекса монаха с наибольшем уровнем ци. Далее, при длине массива. меньшем 2 – массив разбивается на левую и правую часть, 2-м потокам из пула свободных назначается вычисление максимального элемента части переданного массива, если поделенный массив снова окажется длиной, большей 2, то снова образуются две задачи, назначающиеся двум потока. Поток, вызывающий новые, ждет завершения выполнения их задачи. Затем выбирает максимум из 2 значений максимумов левой и правой части и присваивает соответствующий индекс переменной, хранящей номер победителя в массиве. Если в массиве менее 2-х элементов – то возвращается максимальный без образования новых задач для потоков.

При организации работы потоков были использованы параллельные задачи (task) библиотеки OpenMP. Их особенностью является то, что они не порождают потоки, а назначают задачу потоку из пула свободных потоков. При выведении промежуточных данных в консоль у нас появляется общий для потоков объект – консоль, поэтому мы временно закрываем с помощью critical доступ к консоли остальным потокам, пока идет вывод данных одного определенного.

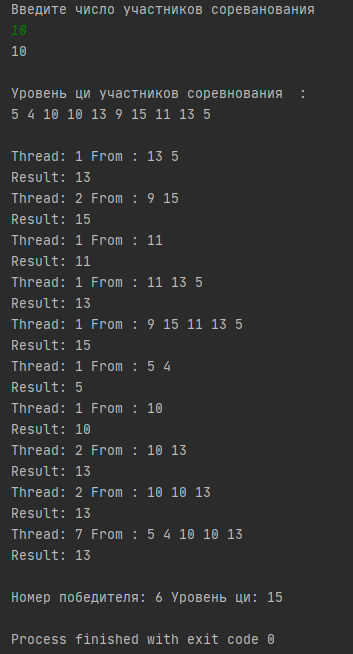
**Тесты**.

Для порождения массива входных данных был использован генератор случайных чисел. От пользователя требуется указать длину массива – целое положительное число. Программа выводит индекс победителя и уровень его энергии ци. Также, для наглядности выводятся промежуточные результаты вычислений в потоках – идентификатор потока, его входной массив и результат вычислений потока. Если есть более одного монаха с одинаковым уровнем ци, то выводится только один.

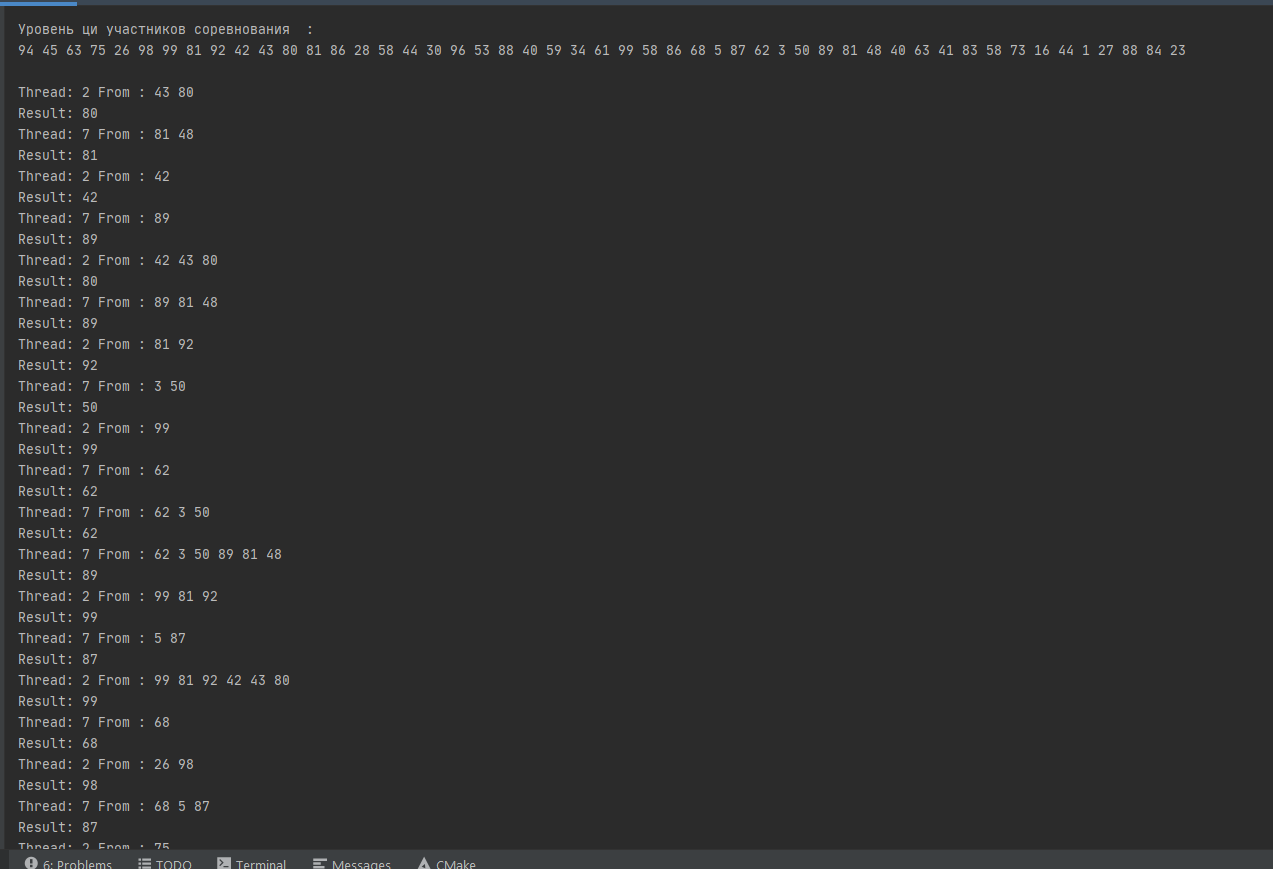
1. n < 1

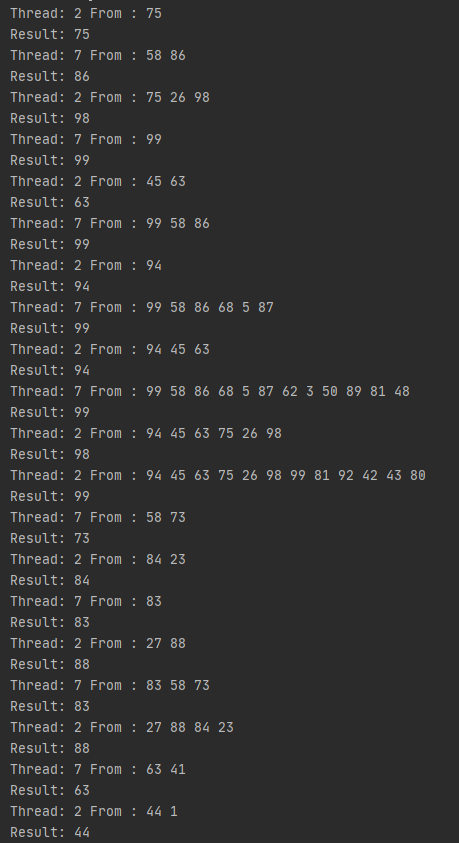


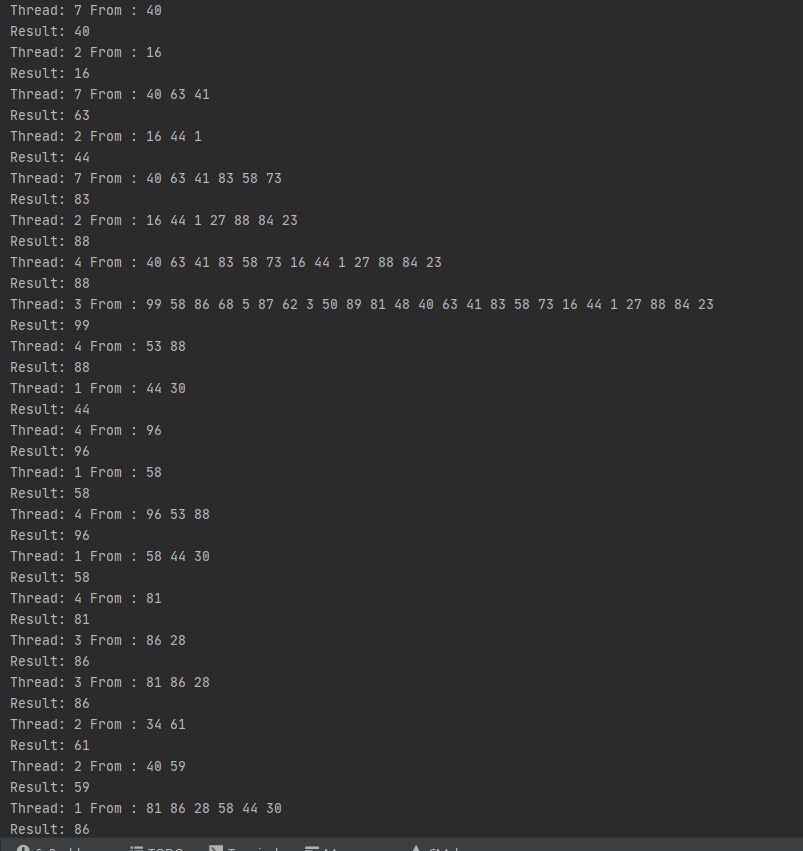
1. n = 10

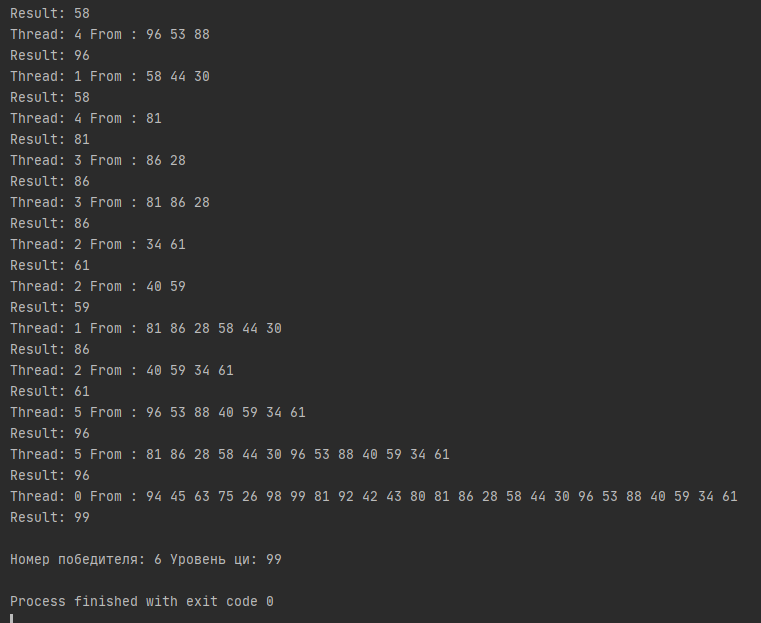


1. n = 50









СПИСОК ИСТОЧНИКОВ

1. [Дихотомия. Википедия.](1.%09https:/ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%85%D0%BE%D1%82%D0%BE%D0%BC%D0%B8%D1%8F#:~:text=%D0%94%D0%B8%D1%85%D0%BE%D1%82%D0%BE%D0%BC%D0%B8%CC%81%D1%8F%20(%D0%B3%D1%80%D0%B5%D1%87.,%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D1%81%D1%8F%20%D0%BD%D0%B0%20%D0%B4%D0%B2%D0%B0%20%D0%B2%D0%B7%D0%B0%D0%B8%D0%BC%D0%BE%D0%B8%D1%81%D0%BA%D0%BB%D1%8E%D1%87%D0%B0%D1%8E%D1%89%D0%B8%D1%85%20%D0%BF%D0%BE%D0%BD%D1%8F%D1%82%D0%B8%D1%8F.)
2. [Блог программиста. Парадигмы параллельного программирования](https://pro-prof.com/forums/topic/parallel-programming-paradigms)
3. [Блог программиста. Учебник по OpenMP](https://pro-prof.com/archives/4335)