НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
Факультет компьютерных наук, Департамент программной инженерии
Курс: «Архитектура вычислительных систем»

Задание 4

Разработка многопоточных приложений с использованием OpenMP

Коновалов Егор Андреевич,

Студент БПИ195

1. Текст задания

Задача об инвентаризации по рядам. После нового года в библиотеке университета обнаружилась пропажа каталога. После поиска и наказания виноватых, ректор дал указание восстановить каталог силами студентов. Фонд библиотека представляет собой прямоугольное помещение, в котором находится M рядов по N шкафов по K книг в каждом шкафу. Требуется создать многопоточное приложение, составляющее каталог. При решении задачи использовать метод «портфель задач», причем в качестве отдельной задачи задается составление каталога одним студентом для одного ряда.

2. Модель вычислений

Приложение использует модель «Взаимодействующие равные» со способом распределения работ «портфель задач».

При такой модели каждый поток сначала берет доступную задачу, получая некоторый номер, указывающий на неё, через глобальную переменную, выполняет её и добавляет результат в массив (вектор), а после, если ещё остались невыполненные задачи , повторяет процесс.

Для разделения доступа к данным, в данном случае к глобальной переменной, содержащей номер доступного для выполнения задания, используется мьютекс.

3. Использованные источники

1. Парадигмы параллельного программирования: https://pro-prof.com/forums/topic/parallel-programming-paradigms
2. Уильямс Энтони. C++. Практика многопоточного программирования. — СПб.: Питер, 2020. — 640 с.

4. Текст программы

```
#include <iostream>
#include <vector>
#include <string>
#include <thread>
#include <stdlib.h>
#include <algorithm>

const int codeLength = 6;
const int threadNumber = 5;

int m = -1, n = -1, k = -1;
int nextTask = 0;
std::vector<std::string> catalog;

std::string getBookCode() {
    int codeValue = rand() % (int)pow(10, codeLength);
    int zeroCount = codeLength - ((int)log10(codeValue) + 1);
```

```
    return std::string(zeroCount, '0') + std::to_string(codeValue);
}

void func() {
    int row;
#pragma omp critical
    {
        row = nextTask++;
        std::thread::id this_id = std::this_thread::get_id();
        std::cout << "row: " << row << " thread: " << this_id << std::endl;
    }

    while (nextTask - 1 < m) {

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < k; j++) {
                std::string record =  getBookCode() + " book"
                        + ", row: " + std::to_string(row + 1)
                        + ", bookcase: " + std::to_string(i + 1);
#pragma omp critical
                {
                    catalog.push_back(record);
                }
            }
        }

#pragma omp critical
        {
            row = nextTask++;
            std::thread::id this_id = std::this_thread::get_id();
            std::cout << "row: " << row << " thread: " << this_id << std::endl;
        }
    }
}

int main() {
    srand(100);

    while (m < 0) {
        std::cout << "Enter number of rows:";
        std::cin >> m;
    }
    while (n < 0) {
        std::cout << "Enter number of bookcases in one row:";
        std::cin >> n;
    }
    while (k < 0) {
        std::cout << "Enter number of books in a bookcases:";
        std::cin >> k;
    }

    catalog.reserve(m * n * k);

#pragma omp parallel for num_threads(threadNumber)
    for (int i = 0; i < threadNumber; ++i) {
        func();
    }

    std::sort(catalog.begin(), catalog.end());

    std::cout << "Number of books: " << std::size(catalog) << std::endl;
    for (auto const& str: catalog) {
```

```
        std::cout << str << std::endl;
    }

    return 0;
}
```