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

Задание 3

Практические приемы построения многопоточных приложений

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 <mutex>

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

std::mutex mutex;

void func() {

    mutex.lock();
    int row = nextTask++;
    mutex.unlock();
```

```
    while (nextTask - 1 < m) {

        std::string record = "row " + std::to_string(row) + "\n";
        for (int i = 0; i < n; i++) {
            record += "\tbookcase " + std::to_string(i) + "\n";
            for (int j = 0; j < k; j++) {
                record += "\t\tbook " + std::to_string(j) + "\n";
            }
        }

        catalog.insert(catalog.begin() + row, record);

        mutex.lock();
        row = nextTask++;
        mutex.unlock();
    }
}

int main() {
    setlocale(LC_ALL, "Russian");

    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.resize(m);

    std::thread t1(func);
    std::thread t2(func);
    t1.join();
    t2.join();

    for (int i = 0; i < m; ++i) {
        std::cout << catalog.at(i) << std::endl;
    }
    return 0;
}
```