ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Департамент программной инженерии

Практические приемы построения многопоточных приложений.

Вариант 25

Исполнитель

Студент группы БПИ195 Пучин М.А.

Задание

Первая задача об Острове Сокровищ. Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Не смотря на добытую карту старого Флинта, местоположение сокровищ попрежнему остается загадкой, поэтому искать клад приходится практически на ощупь. Так как Сильвер ходит на деревянной ноге, то самому бродить по джунглям ему не с руки. Джон Сильвер поделил остров на участки, а пиратов на небольшие группы. Каждой группе поручается искать клад на одном из участков, а сам Сильвер ждет на берегу. Пираты, обшарив свою часть острова, возвращаются к Сильверу и докладывают о результатах. Требуется создать многопоточное приложение с управляющим потоком, моделирующее действия Сильвера и пиратов.

Состав программы

Программа представляет собой симулятор действий Сильвера (поток silver), создающий потоки group, которые ищут на острове, представленном в виде булевого массива, сокровища, которые выглядят как отмеченные true клетки (индекс генерируется рандомно).

После запуска программа ожидает на вход 2 числа в одну строку (первое – количество групп пиратов, второе – длина отрезка, который обыскивает каждая группа).

В программе выделены функции:

manage – функция создания потоков групп пиратов и управления ими.

startThread – функция поиска группой пиратов сокровища на отрезке l, r.

Текст программы:

```
#include <iostream>
#include <utility>
#include <vector>
#include <mutex>

//Вариант 25

using namespace std;

vector<bool> field;
```

```
pair<int, int> winner = { -1, -1 };
mutex mtx;

void startThread(int id, int l, int r)
{
    for (int i = l; i < r; i++) {
        if (field[i] == true) {
            winner = { id, i };
            mtx.lock();
            cout << "Group " << id << " " << " found treasures!" << endl;
            mtx.unlock();
            return;
        }
    }
    mtx.lock();
    cout << "Group " << id << " " << " found nothing" << endl;
    mtx.unlock();
    return;
}

void manage(int num, int area) {
    thread *thread_array = new thread[num] ;
    for (int i = 0; i < num; i++) {
        thread_array[i] = thread(startThread, i, i * area, i * area + area);
    }
    for (int i = 0; i < num; i++) {
        if (thread_array[i].joinable()) {
            thread_array[i].join();
        }
    }
    cout << "Found in " << winner.second << " by group " << winner.first << endl;
}

signed main() {
    int number_of_groups, area;
    cin >> number_of_groups >> area;
    field.resize(number_of_groups * area);
    srand(4);
    int treasure = rand() % field.size();
    field[treasure] = true;
    thread silver(manage, number_of_groups, area);
    silver.join();
}
```