# Домашнее задание №4.

# Вторая задача об Острове Сокровищ.

**Штырба Владислав Анатольевич**

**НИУ ВШЭ**

**Факультет Компьютерных наук**

**Группа БПИ197**

## Текст задания

26. Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Несмотря на добытую карту старого Флинта, местоположение сокровищ по- 8 прежнему 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 нескольких участках, а сам Сильвер ждет на берегу. Группа пиратов, обшарив одну часть острова, переходит к другой, еще необследованной части. Закончив поиски, пираты возвращаются к Сильверу и докладывают о результатах. Требуется создать многопоточное приложение с управляющим потоком, моделирующее действия Сильвера и пиратов. При решении использовать парадигму портфеля задач.

## Применяемые расчётные методы

Потоки предоставляют возможность проведения параллельных или псевдопараллельных (в случае одного ядра) вычислений. Потоки могут порождаться во время работы программы, процесса или другого потока. Основное отличие потоков от процессов заключается в том, что различные потоки имеют различные пути выполнения, но при этом пользуются общей памятью. Таким образом, несколько порожденных в программе потоков, могут пользоваться глобальными переменными, и любое изменение данных одним потоком, будет доступно и для всех остальных. Существует несколько моделей построения многопоточных приложений.

## Исходный код

```
#include <iostream>
#include <thread>
#include <math.h>
#include <chrono>
#include <cstdlib>
```

```cpp
#include <ctime>
#include <string>
#include <vector>
#include <omp.h>

using namespace std;
using namespace std::chrono;

int result;

void crewFind(bool* islandPart, int size, int islandI) {
    for (int i = 0; i < size; i++)
        if (islandPart[i])
            result = islandI * size + i;
}

void find(int crewN, bool* island, int size) {
    auto startTime = high_resolution_clock::now();

    std::cout << "# of threads = " << crewN << endl;
    // thread is a crew group
    int tileCrN = ceil(size * 1.0 / crewN);

    bool** islandParts = new bool* [crewN];
    for (int i = 0; i < crewN; i++)
        islandParts[i] = new bool[tileCrN];

    int begin = 0;
    int end = tileCrN;

    for (int i = 0; i < crewN; i++) {
        for (int j = begin; j < end; j++) {
            if (j >= size)
                islandParts[i][j - begin] = false;
            else islandParts[i][j - begin] = island[j];
        }
        begin = end;
        end = begin + tileCrN;
    }

    for (int i = 0; i < crewN; i++) {
        for (int j = 0; j < tileCrN; j++)
            std::cout << (islandParts[i][j] ? "1 " : "0 ");
        std::cout << endl;
    }
    int i = 0;
    thread* crews = new thread[crewN];
  /*  while (true)
    {
        auto t = thread(crewFind, islandParts[i], tileCrN, i);
        t.join();
        if (i == crewN - 1)
        {
            break;
        }
        i++;
    }*/

    omp_set_num_threads(crewN);
#pragma omp parallel
    {
        auto num = omp_get_num_threads();
#pragma omp for
        {
            for (int i = 0; i < num; i++)
```

```
                crews[i] = thread(crewFind, islandParts[i], tileCrN, i);
        }
#pragma omp for
        {
            for (int i = 0; i < num; i++)
                crews[i].join();
        }
    }

    auto stopTime = high_resolution_clock::now();
    auto duration = duration_cast<microseconds>(stopTime - startTime);

    std::cout << "Found treasure at " << result << endl;
    std::cout << "Spent time: \t" << duration.count() * 1.0 / 1000000 << endl;

    for (int i = 0; i < crewN; i++)
        delete[] islandParts[i];
    delete[] islandParts;
}

void print(bool* island, int size) {
    for (int i = 0; i < size; i++)
        std::cout << to_string(i) << ") " << (island[i] ? "1" : "0") << endl;
    std::cout << "\n";
}

bool check(string line) {
    for (int i = 0; i < line.length(); i++)
        if (line[i] < '0' || line[i] > '9')
            return false;
    return true;
}

int main()
{
    srand(time(NULL));
    string line;
    std::cout << "Enter the size of the island: ";
    cin >> line;
    if (!check(line)) {
        std::cout << "Enter a positive integer pls" << endl;
        return 0;
    }
    int islandSize = stoi(line);
    bool* island = new bool[islandSize] {};

    island[rand() % islandSize] = true;

    print(island, islandSize);

    for (int i = 1; i < 9; i++) {
        find(i, island, islandSize);
        std::cout << this_thread::get_id() << endl;
        std::cout << "\n";
    }

    delete[] island;
    return 0;
}
```

# Тест программы

Тест с неверными введёнными данными: test1_wrong_input.png

Тест с верным вводом: test2.png