**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное**

**учреждение высшего образования «Национальный исследовательский университет «Высшая школа экономики»**

Факультет компьютерных наук

Департамент программной инженерии

**Отчёт**

**к домашнему заданию №3 по дисциплине**

**«Архитектура вычеслительных систем»**

Работу выполнил:

Студент 2 курса группы БПИ194

1 подгруппы

Ткаченко Эдуард Витальевич

Задание

Шайка пиратов под предводительством Джона Сильвера высадилась на берег Острова Сокровищ. Не смотря 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.

Решение

Представим карту сокровищ как таблицу с типом bool из n строк и m столбцов. Если в значение элемента таблицы – true, то значит данный элемент обозначает клад. Будем использовать парадигму «портфель задач», о котором подробно написано в «Основы многопоточного, параллельного и распределенного программирования» [1]. Задачей будет являться поиск сокровища в очередной строке. Будем искать сокровище, пробегая по строкам. Когда какая-нибудь команда(поток) завершит поиск сокровища в данной ей строке (или она ещё не искала сокровища) и ещё не все строки просмотрены, команда возьмет новую задачу(создан новый поток). Состояния гонки не будет, так как сокровище единственное.

Входные данные

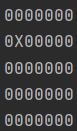
В консоль вводятся следующие входные данные:

1. Размер карты по высоте.
2. Размер карты по ширине.
3. Количесво групп пиратов.

Выходные данные

В консоль выводятся следующие выходные данные:

1. Карта сокровищ в виде таблицы, где в i-ой строке j-ом столбце ‘0’ обозначает, что в ячейке нет сокровища, “X” – есть сокровище. Пример:



1. Координаты сокровища (нумерация с 0).

Текст программы

// Tkachenko Eduard

// Вариант 26

#include <cassert>

#include <iostream>

#include <vector>

#include <thread>

// Поиск в строке nmbI контейнера map местоположения элемента со значением true.

void find(std::vector<std::vector<bool>>& map, int nmbI, int &ans\_i, int &ans\_j, char &finish) {

for (int j = 0; j < map[nmbI].size(); ++j) {

if (map[nmbI][j]) {

ans\_i = nmbI;

ans\_j = j;

finish = 0;

return;

}

}

finish = 0;

}

int main() {

int n; // Размер карты во высоте.

int m; // Размер карты по ширине.

int k; // Количество потоков (команд Флинта).

std::cout << "Map size in height:";

std::cin >> n;

std::cout << "Map size in width:";

std::cin >> m;

std::cout << "Number of teams:";

std::cin >> k;

if (k <= 0 || n <= 0 || m <= 0) {

std::cout << "Incorrect input.";

return 0;

}

std::vector<std::vector<bool>> map(n, std::vector<bool>(m)); // Карта сокровищ.

map[rand() % n][rand() % m] = true; // Генерация координат сокровищ.

int ans\_i = -1, ans\_j = -1; // Местоположение сокровищ

int curI = 0; // Номер строки для вызова find очередным потоком.

std::vector<std::thread> threads(k); // Возможные потоки

std::vector<char> is\_run(k); // Отслеживание работы потоков

while (ans\_i == -1) {

for (int i = 0; i < threads.size(); ++i) {

if (!is\_run[i] && curI != n) {

if (threads[i].joinable())

threads[i].join();

is\_run[i] = 1;

threads[i] = std::thread(find, std::ref(map), curI++, std::ref(ans\_i), std::ref(ans\_j), std::ref(is\_run[i]));

}

}

}

for (int i = 0; i < k; ++i) {

if (threads[i].joinable())

threads[i].join();

}

std::cout << "Treasure map: \n";

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

std::cout << (map[i][j] ? 'X' : '0'); // 'X' - сокровище, '0' - нет сокровища

}

std::cout << std::endl;

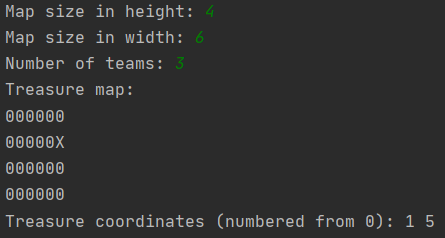
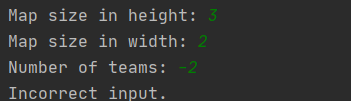
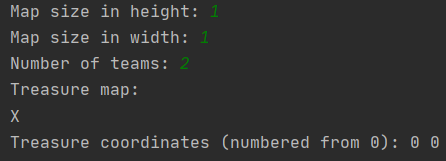
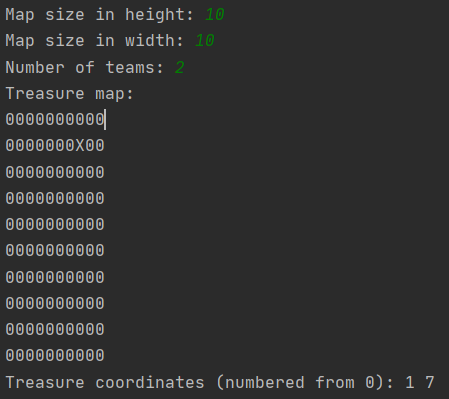
}

std::cout << "Treasure coordinates (numbered from 0): " << ans\_i << ' ' << ans\_j;

return 0;

}

Тестирование

1. 
2. 
3. 
4. 

Список использованных источников

1. Эндрюс Г.Р. Основы многопоточного, параллельного и распределенного программирования. Пер. сангл. — М. : Издательский дом “Вильямс”, 2003. — 512 с. : ил.