**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное**

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

Факультет компьютерных наук

Департамент программной инженерии

# Отчет к домашнему заданию по дисциплине

**«Архитектура вычислительных систем»**

Работу выполнил:

Студент группы БПИ195 Цуркан Даниел

**Москва 2020**

**Задание (Вариант – 29, задача - 7)**

Анчуария и Тарантерия – два крохотных латиноамериканских гос-ва, затерянных в южных Андах. Диктатор Анчуарии, дон Федерико, объявил войну диктатору Тарантерии, дону Эрнандо. У обоих диктаторов очень мало солдат, но очень много снарядов для минометов, привезенных с последней американской гуманитарной помощью. Поэтому армии обеих сторон просто обстреливают наугад территорию противника, надеясь поразить что-нибудь ценное. Стрельба ведется по очереди до тех пор, пока либо не будут уничтожены все цели, либо стоимость потраченных снарядов не превысит суммарную стоимость всего того, что ими можно уничтожить. Создать многопоточное приложение, моделирующее военные действия.

**Решение**

В начале программа просит пользователя ввести необходимые значения. Программа создает территорию двух противников и потоки, имитирующие минометы. Далее программа имитирует сражение до тех пор, пока соблюдаются некоторые условия. Когда одно из условий оказывается неудовлетворяющим условию ведения войны, мы выводим информацию о победителе.

Решение задачи было разбито на несколько функций: Zone:: attack(int i, int j), Zone:: print(), Zone:: isAlive(), Zone:: getAllCost(), mortar(int id, std::string team, Zone\* enemy, int randKey), readNumber(int minVal, int maxValue).

Функция Zone:: attack(int i, int j) атакует переданный в качестве параметров сектор.

Функция Zone:: print() выводит созданную территорию на экран.

Функция Zone:: isAlive() проверяет боеспособно ли еще гос-во.

Функция Zone:: getAllCost() проверяет не превысила ли стоимость потраченных снарядов суммарную стоимость всего того, что ими уничтожено.

Функция mortar(int id, std::string team, Zone\* enemy, int randKey) имитирует стрельбу из одного миномета.

Функция readNumber(int minVal, int maxValue) просит пользователя ввести число в указанном диапазоне.

**Текст программы**

#include <iostream>

#include <mutex>

#include <thread>

#include <condition\_variable>

#pragma ide diagnostic ignored "EndlessLoop"

int const costOfBullet = 10; //Стоимость одного снаряда

int const costOfObj = 100; //Стоимость одного объекта

int const sizeZone = 10; //Размер поля

int const minCountOfObjectsOnLine = 1; //Минимальное количество объектов на одной линии

int const maxCountOfObjectsOnLine = sizeZone; //Максимальное количество объектов на одной линии

int const minLoadTime = 1000; //Минимальное время зарядки орудия (мс)

int const maxLoadTime = 3000; //Максимальное время зарядки орудия (мс)

int const minDelayTime = 100; //Минимальная задержка при выстреле (мс)

int const maxDelayTime = 300; //Максимальное задерэка при выстреле (мс)

std::condition\_variable cv;

std::mutex show; //мьютекс для корректного отображения сообщений

std::mutex volley; //мьютекс залпа для условной переменной

int countLoaded = 0; //количество заряженных орудий

bool startVolley; //начался ли залп

class Zone {

private:

bool\*\* zone;

std::mutex zoneMtx;

std::string name;

int countOfObjects;

int startCountOfObjects;

public:

/\*\*

\* Создает зону

\* @param name имя зоны

\*/

Zone(std::string name) {

startCountOfObjects = 0;

zone = new bool\*[sizeZone];

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

zone[i] = new bool[sizeZone];

int countOfObjects = 0;

int countObjectsOnLine = rand() % (maxCountOfObjectsOnLine - minCountOfObjectsOnLine)

+ minCountOfObjectsOnLine;

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

if (countOfObjects < countObjectsOnLine && rand() % 2 == 0) {

zone[i][j] = true;

countOfObjects++;

} else

zone[i][j] = false;

}

startCountOfObjects += countOfObjects;

this->countOfObjects = startCountOfObjects;

}

this->name = name;

}

/\*\*

\* Атакует переданный сектор

\* @param i

\* @param j

\* @return

\*/

bool attack(int i, int j) {

zoneMtx.lock();

bool value = zone[i][j];

zoneMtx.unlock();

if (value) {

zoneMtx.lock();

zone[i][j] = false;

zoneMtx.unlock();

countOfObjects--;

show.lock();

printf("[%s] Object in sector (%d, %d) was destroyed\n", name.c\_str(), i, j);

show.unlock();

} else {

show.lock();

printf("[%s] There was nothing in sector (%d, %d)\n", name.c\_str(), i, j);

show.unlock();

}

return value;

}

/\*\*

\* Выводит зону на экран

\*/

void print() {

std::cout << name << ":" << std::endl;

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

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

if (zone[i][j])

std::cout << "o ";

else

std::cout << "# ";

}

std::cout << "\n";

}

}

bool isAlive() {

return countOfObjects > 0;

}

int getAllCost() {

return startCountOfObjects \* costOfObj;

}

};

void mortar(int id, std::string team, Zone\* enemy, int randKey) {

srand(randKey);

//Занимает позицию

std::this\_thread::sleep\_for(std::chrono::milliseconds(rand() % 3000 + 4000));

show.lock();

printf("[%s : Martian %d] Took position!\n", team.c\_str(), id);

show.unlock();

//Заряжает орудие

std::this\_thread::sleep\_for(std::chrono::milliseconds(rand() % (maxLoadTime - minLoadTime) + minLoadTime));

show.lock();

printf("[%s : Martian %d] The gun is loaded!\n", team.c\_str(), id);

countLoaded++;

show.unlock();

while (true) {

//Ждет приказа стрелять

std::unique\_lock<std::mutex> volleyLock(volley);

while (!startVolley)

cv.wait(volleyLock);

volleyLock.unlock();

//Выбирает сектор для обстрела

int x = rand() % sizeZone;

int y = rand() % sizeZone;

show.lock();

printf("[%s : Martian %d] Attacking zone (%d, %d)\n", team.c\_str(), id, x, y);

show.unlock();

std::this\_thread::sleep\_for(std::chrono::milliseconds(rand() % (maxDelayTime - minDelayTime) + minDelayTime));

enemy->attack(x, y);

//Заряжает орудие

std::this\_thread::sleep\_for(std::chrono::milliseconds(rand() % (maxLoadTime - minLoadTime) + minLoadTime));

show.lock();

printf("[%s : Martian %d] The gun is loaded!\n", team.c\_str(), id);

countLoaded++;

show.unlock();

}

}

int readNumber(int minVal, int maxValue) {

int number;

std::cin >> number;

while (number < minVal && number > maxValue) {

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

std::cin >> number;

}

return number;

}

int main() {

srand(time(0));

std::cout << "Enter count of mortars in first army:";

int countOfMortarsFirstArmy = readNumber(1, 50);

std::cout << "Enter count of mortars in second army:";

int countOfMortarsSecondArmy = readNumber(1, 50);

//Создаем и выводим поля

Zone zone1("Anchuria");

Zone zone2("Taranttery");

zone1.print();

zone2.print();

//Создаем потоки

std::thread\* mortars1 = new std::thread[countOfMortarsFirstArmy];

std::thread\* mortars2 = new std::thread[countOfMortarsSecondArmy];

std::cout << "[Main Thread] Prepare your weapons for the battle!\n";

for (int i = 0; i < countOfMortarsFirstArmy; ++i)

mortars1[i] = std::thread(mortar, i + 1, "Anchuria", &zone2, rand());

for (int i = 0; i < countOfMortarsSecondArmy; ++i)

mortars2[i] = std::thread(mortar, i + 1, "Taranttery", &zone1, rand());

int cost1 = 0;

int cost2 = 0;

//Начинаем войну до того момента пока кто-то не победит или стоимось снарядов не привысит

//Стоимости варжеских сооружений

while (zone1.isAlive() && zone2.isAlive() &&

cost1 < zone2.getAllCost() && cost2 < zone1.getAllCost()) {

while (countLoaded < countOfMortarsSecondArmy + countOfMortarsFirstArmy)

std::this\_thread::sleep\_for(std::chrono::milliseconds(100));

countLoaded = 0;

show.lock();

printf("[Main Thread] Fire!!!\n");

show.unlock();

std::this\_thread::sleep\_for(std::chrono::milliseconds(1000));

startVolley = true;

cv.notify\_all();

std::this\_thread::sleep\_for(std::chrono::milliseconds(500));

startVolley = false;

cost1 += costOfBullet \* countOfMortarsFirstArmy;

cost2 += costOfBullet \* countOfMortarsSecondArmy;

show.lock();

zone1.print();

zone2.print();

printf("Anchuria bullets cost: %d; Taranttery Objects Cost: %d\nTaranttery bullets cost: %d; Anchuria Objects Cost: %d\n", cost1,

zone2.getAllCost(), cost2, zone1.getAllCost());

show.unlock();

}

//Подводим итоги битвы

if (zone1.isAlive() && zone2.isAlive())

std::cout << "The war is too expensive...\n";

else if (zone2.isAlive())

std::cout << "Taranttery win!!!\n";

else

std::cout << "Anchuria win!!!\n";

for (int i = 0; i < countOfMortarsFirstArmy; ++i)

mortars1[i].detach();

for (int i = 0; i < countOfMortarsSecondArmy; ++i)

mortars2[i].detach();

delete[] mortars1;

delete[] mortars2;

return 0;

}

**Тестирование**

Результат выполнения программы приведен на рисунке 1.

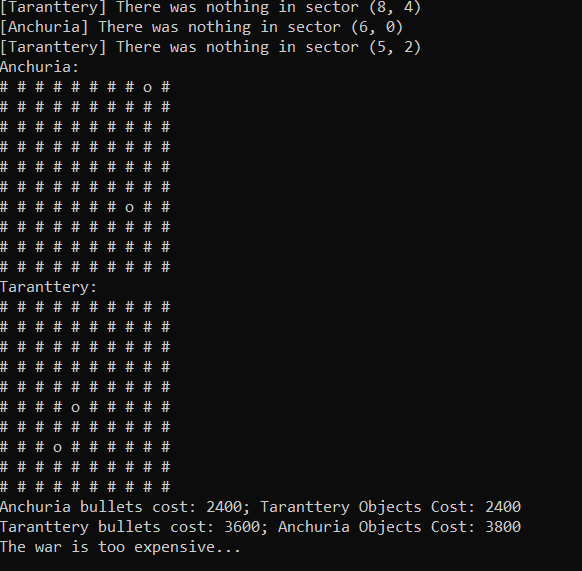


Рисунок 1 – Результат выполнения программы

**Список используемых источников**

1. SoftCraft разноликое программирование «Программирование на языке ассемблера. Микропроект. Требования к оформлению. 2020-2021 уч.г.» (<http://softcraft.ru/edu/comparch/tasks/mp01/>)
2. https://docs.microsoft.com/ru-ru/cpp/build/walkthrough-compiling-a-native-cpp-program-on-the-command-line?view=msvc-160