

Чернівецький національний університет імені Юрія Федьковича  
Навчально-науковий інститут фізико-технічних та комп'ютерних наук  
Відділ комп'ютерних технологій  
Кафедра математичних проблем управління і кібернетики

Звіт  
про виконання лабораторної роботи №2

Тема: “ Багатопоточне програмування.”

з дисципліни  
“ Літня обчислювальна практика”

Варіант № 12 (2)

Виконав:  
ст. гр. 241 Подарунок М.  
Прийняв:  
доц. Лазорик В. В.

Чернівці – 2025

## Практика. Лабораторна робота № 2.

Мета роботи: Ознайомитися з багатопоточним програмуванням.

### Виконання лабораторної роботи

1. Зайти в свій обліковий запис на [github.com](https://github.com). Зайти в [github classroom](https://github.com/classroom).
2. Клонувати репозиторій <https://classroom.github.com/a/5KirRs5C> в свій обліковий запис на [github.com](https://github.com).
3. Розв'язати завдання.
4. Вихідних код записати в створений репозиторій.

### Завдання для лабораторної роботи

#### Варіант 2

1. Бджоли-робочі рухаються в один з кутів області їх проживання (наприклад,  $[0; 0]$ ) по прямій з швидкістю  $V$ , а потім повертатися назад в точку свого народження з тією ж швидкістю.
2. Трутні рухаються хаотично зі швидкістю  $V$ . Хаотичність досягається випадкової зміною напрямку руху раз в  $N$  секунд

```

    }
    else {
        // Переміщення на крок у напрямку цілі
        x += V * dx / dist;
        y += V * dy / dist;
    }

    // Вивід позиції (із захистом м'ютексом)
    {
        std::lock_guard<std::mutex> lock(cout_mutex);
        cout << "[WorkerBee] Position: (" << x << ", " << y << ")" << endl;
    }

    // Пауза 500 мс
    this_thread::sleep_for(chrono::milliseconds(500));
}
};

// Клас трутня
class Drone {
    float x, y; // позиція
    float dirX, dirY; // напрям руху
    std::mt19937 rng; // генератор випадкових чисел
    std::uniform_real_distribution<float> angle_dist; // розподіл кутів

public:
    Drone(float startX, float startY)
        : x(startX), y(startY), rng(random_device{}()),
        angle_dist(0, 2 * M_PI) {
        changeDirection(); // ініціалізація напрямку
    }

    // Випадкова зміна напрямку
    void changeDirection() {
        float angle = angle_dist(rng);
        dirX = cos(angle);
        dirY = sin(angle);
    }

    void move() {
        int counter = 0;
        while (true) {
            // Зміна напрямку кожні N секунд (~N*2 кроки по 500мс)
            if (counter % (N * 2) == 0) {

```

```

#include <iostream>
#include <thread>
#include <vector>
#include <cmath>
#include <random>
#include <chrono>
#include <mutex>

using namespace std;

#ifndef M_PI
#define M_PI 3.14159265358979323846
#endif

// Глобальні параметри
const float V = 1.0f;           // швидкість руху
const int N = 3;                // період зміни напрямку для трутнів (в секундах)
const int SIM_DURATION = 20;    // тривалість симуляції (секунди)

// М'ютекс для синхронізації виводу в консоль
std::mutex cout_mutex;

// Клас бджоли-робочої
class WorkerBee {
public:
    WorkerBee(float startX, float startY)
        : x(startX), y(startY), origin_x(startX), origin_y(startY) {}

    void move() {
        while (true) {
            // Обчислення вектора напрямку
            float dx = (to_zero ? -x : origin_x - x);
            float dy = (to_zero ? -y : origin_y - y);
            float dist = sqrt(dx * dx + dy * dy);

            // Якщо близько до цілі – змінюємо напрямок
            if (dist < V) {
                x = (to_zero ? 0 : origin_x);
                y = (to_zero ? 0 : origin_y);
                to_zero = !to_zero;
            }
        }
    }
};

```

```

        changeDirection();
    }

    // Переміщення
    x += V * dirX;
    y += V * dirY;

    // Вивід позиції
    {
        std::lock_guard<std::mutex> lock(cout_mutex);
        cout << "[Drone] Position: (" << x << ", " << y << ")" << endl;
    }

    ++counter;
    this_thread::sleep_for(chrono::milliseconds(500));
}

};

int main() {
    vector<thread> threads;

    // Створюємо 3 бджоли-робочі
    for (int i = 0; i < 3; ++i) {
        WorkerBee bee(10.0f + i * 5, 10.0f + i * 5);
        threads.emplace_back(&WorkerBee::move, bee); // потік викликає метод move()
    }

    // Створюємо 2 трутнів
    for (int i = 0; i < 2; ++i) {
        Drone drone(20.0f + i * 5, 20.0f + i * 5);
        threads.emplace_back(&Drone::move, drone);
    }

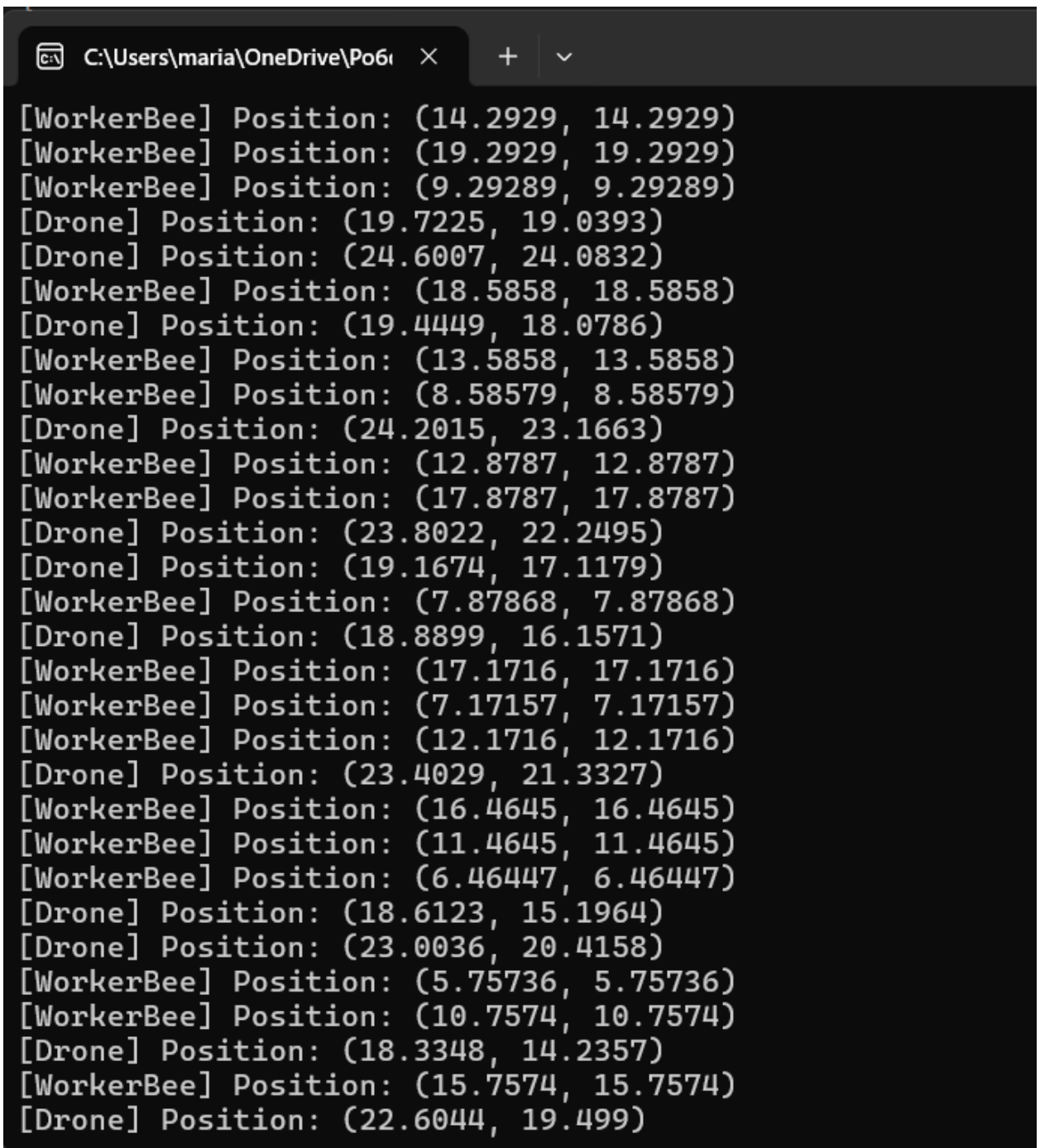
    // Очікуємо завершення симуляції
    this_thread::sleep_for(chrono::seconds(SIM_DURATION));

    {
        std::lock_guard<std::mutex> lock(cout_mutex);
        cout << "Simulation complete. Press Ctrl+C to terminate." << endl;
    }

    return 0; // Програма не завершує потоки (для демонстрації)
}

```

Рис. 1 – Код завдання



```
C:\Users\maria\OneDrive\Рабочий стол\... × + ▾  
[WorkerBee] Position: (14.2929, 14.2929)  
[WorkerBee] Position: (19.2929, 19.2929)  
[WorkerBee] Position: (9.29289, 9.29289)  
[Drone] Position: (19.7225, 19.0393)  
[Drone] Position: (24.6007, 24.0832)  
[WorkerBee] Position: (18.5858, 18.5858)  
[Drone] Position: (19.4449, 18.0786)  
[WorkerBee] Position: (13.5858, 13.5858)  
[WorkerBee] Position: (8.58579, 8.58579)  
[Drone] Position: (24.2015, 23.1663)  
[WorkerBee] Position: (12.8787, 12.8787)  
[WorkerBee] Position: (17.8787, 17.8787)  
[Drone] Position: (23.8022, 22.2495)  
[Drone] Position: (19.1674, 17.1179)  
[WorkerBee] Position: (7.87868, 7.87868)  
[Drone] Position: (18.8899, 16.1571)  
[WorkerBee] Position: (17.1716, 17.1716)  
[WorkerBee] Position: (7.17157, 7.17157)  
[WorkerBee] Position: (12.1716, 12.1716)  
[Drone] Position: (23.4029, 21.3327)  
[WorkerBee] Position: (16.4645, 16.4645)  
[WorkerBee] Position: (11.4645, 11.4645)  
[WorkerBee] Position: (6.46447, 6.46447)  
[Drone] Position: (18.6123, 15.1964)  
[Drone] Position: (23.0036, 20.4158)  
[WorkerBee] Position: (5.75736, 5.75736)  
[WorkerBee] Position: (10.7574, 10.7574)  
[Drone] Position: (18.3348, 14.2357)  
[WorkerBee] Position: (15.7574, 15.7574)  
[Drone] Position: (22.6044, 19.499)
```

Рис. 2 – Результат виконання завдання

Висновок: Ми ознайомилися з багатопоточним програмуванням.