НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет прикладної математики
Кафедра прикладної математики

Звіт
із лабораторної роботи №2
із дисципліни «Розподілені і хмарні обчислення»

Виконав:

студент групи КМ-01

Романецький М.С.

Керівник:

Доцент кафедри ПМА
Лискін В. О.

Київ — 2023

Мета роботи: Порівняти однопоточну та багатопоточну версії матричного множення та додавання елементів матриці

Опис програми: Бібліотека ‘Rayon’ буде використовуватись для паралелізму. Для порівняння часу роботи буде використовувати ‘Criterion’

Процесор	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz
ОЗП	8.00 ГБ (доступно для використання: 7.69 ГБ)

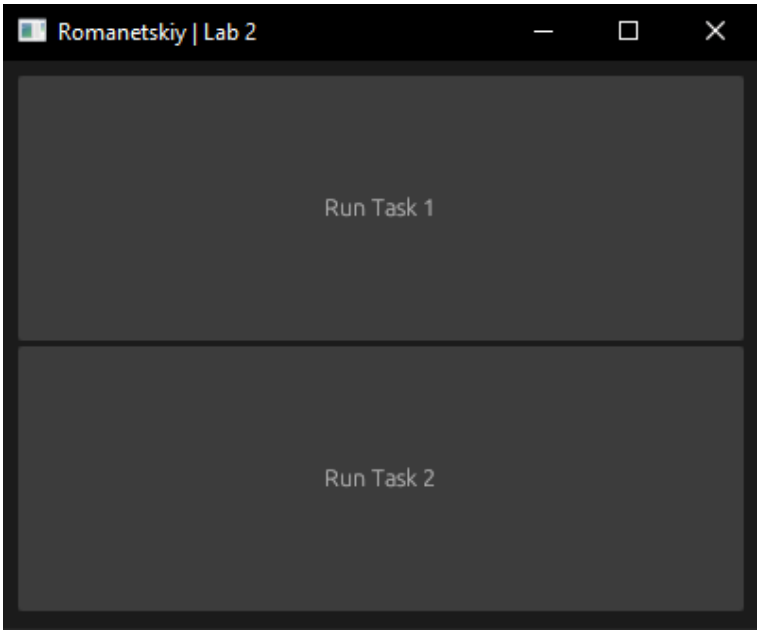
Характеристики системи
Кількість ядер процесора = 8

		Середній час
Множення матриць	Однопоточна реалізація	11.295 ms
	Багатопоточна реалізація	3.1764 ms
Додавання матриць	Однопоточна реалізація	4.6789 ms
	Багатопоточна реалізація	1.3938 ms

Отже багатопоточна реалізація в даному випадку працює швидше, іноді в декілька разів

```
Task 1:
CPU cores number: 8
1 Hello World !
2 Hello World !
3 Hello World !
4 Hello World !
5 Hello World !
6 Hello World !
7 Hello World !
8 Hello World !
```

Завдання 1



Інтерфейс програми

Код програми:

```
// cd /d/KPI/Distributed_computing/Labs/lab_2


use eframe::{epi, egui::{self, CtxRef}};
use rayon::{current_num_threads};
use std::process::Command;
use egui::vec2;

struct MyApp {
    //  дані та стан програми
}

impl Default for MyApp {
    fn default() -> Self {
        Self {
            // Ініціалізація стану
        }
    }
}

impl epi::App for MyApp {
    fn name(&self) -> &str {
        "Romanetskiy | Lab 2"
    }

    fn update(&mut self, ctx: &CtxRef, _frame: &mut epi::Frame) {
        egui::CentralPanel::default().show(ctx, |ui| {
            let button_size = vec2(ui.available_width(), 140.0);
            if ui.add_sized(button_size, egui::Button::new("Run Task
1")).clicked() {
                task_1()
            }
            if ui.add_sized(button_size, egui::Button::new("Run Task
2")).clicked() {
                task_2()
            }
        });
    }
}


fn main() {
    let app = MyApp::default();
    let mut native_options = eframe::NativeOptions::default();
    native_options.initial_window_size = Some(egui::vec2(400.0, 300.0));
    eframe::run_native(Box::new(app), native_options);
}
```

```rust
fn task_1() {
    println!("\nTask 1:");
    let num_cores: usize = current_num_threads();
    println!("CPU cores number: {num_cores}");
    for i in 0..num_cores {
        let n = i + 1;
        println!("{n} Hello World !");
    }
}


fn task_2() {
    println!("\nTask 2:");
    println!("Запускаємо cargo bench");
    match Command::new("cargo").args(&["bench"]).status() {
        Ok(status) => if status.success() {
            println!("cargo bench виконано успішно");
        } else {
            eprintln!("cargo bench завершилося з помилкою");
        },
        Err(e) => eprintln!("Помилка при запуску cargo bench: {}", e),
    }
}
```