Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Основи паралельного програмування

Лабораторна робота №8

**«Потоки в Rust»**

Виконала:

студентка групи ІО-64

Бровченко А. В.

Перевірив:

Корочкін О. В.

Київ

2018 р.

**Завдання:**

F1: C = A – B\*(MA\*MС)\*е

F2: MF = MF\*MG\*k

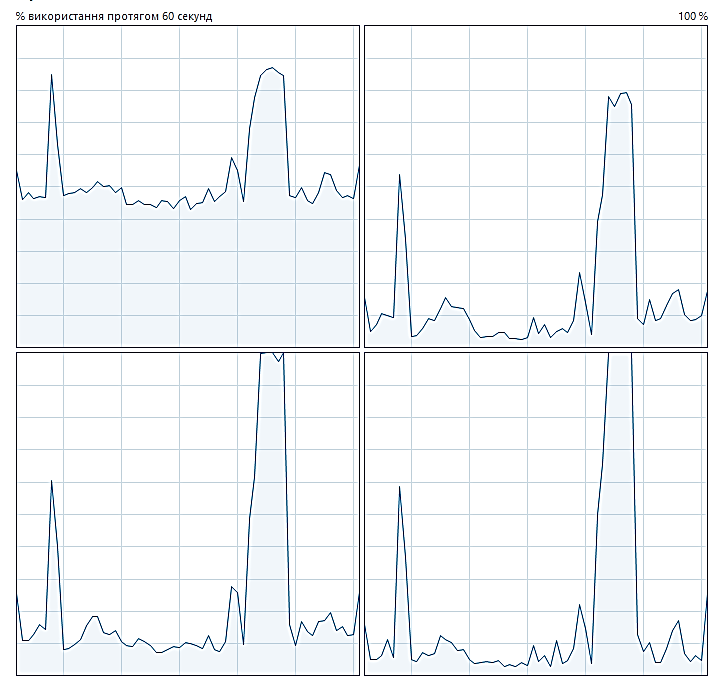
F3: O = SORT(P)\*(MR\*MT)

**Лістинг програми:**

**Lab8.rs**

use std::thread;  
use rulinalg::matrix::Matrix;  
use rand::Rng;  
  
fn f1(size: usize) -> Matrix<f64> {  
 let mut rng = rand::thread\_rng();  
  
 let e: f64 = rng.gen();  
 let mut a\_vec: Vec<f64> = Vec::new();  
 let mut b\_vec: Vec<f64> = Vec::new();  
 let mut ma\_vec: Vec<f64> = Vec::new();  
 let mut mc\_vec: Vec<f64> = Vec::new();  
  
 for \_ in 0..size {  
 a\_vec.push(rng.gen());  
 b\_vec.push(rng.gen());  
 for \_ in 0..size {  
 ma\_vec.push(rng.gen());  
 mc\_vec.push(rng.gen());  
 }  
 }  
 let a = Matrix::new(1, size, a\_vec);  
 let b = Matrix::new(1, size, b\_vec);  
 let ma = Matrix::new(size, size, ma\_vec);  
 let mc = Matrix::new(size, size, mc\_vec);  
 // println!("A = \n{},\nB = \n{},\nMA = \n{},\nMC = \n{},\ne = {}", a, b, ma, mc, e);  
 // println!("F1 = {}", result);  
 a - (b \* (ma \* mc) \* e)  
}  
  
fn f2(size: usize) -> Matrix<f64> {  
 let mut rng = rand::thread\_rng();  
  
 let k: f64 = rng.gen();  
 let mut mf\_vec: Vec<f64> = Vec::new();  
 let mut mg\_vec: Vec<f64> = Vec::new();  
  
 for \_ in 0..size {  
 for \_ in 0..size {  
 mf\_vec.push(rng.gen());  
 mg\_vec.push(rng.gen());  
 }  
 }  
 let mf = Matrix::new(size, size, mf\_vec);  
 let mg = Matrix::new(size, size, mg\_vec);  
 // println!("MF = \n{},\nMG = \n{},\nk = {}", mf, mg, k);  
 // println!("F2 = {}", result);  
 mf \* mg \* k  
}  
  
fn f3(size: usize) -> Matrix<f64> {  
 let mut rng = rand::thread\_rng();  
  
 let mut p: Vec<f64> = Vec::new();  
 let mut mr\_vec: Vec<f64> = Vec::new();  
 let mut mt\_vec: Vec<f64> = Vec::new();  
  
 for \_ in 0..size {  
 p.push(rng.gen());  
 for \_ in 0..size {  
 mr\_vec.push(rng.gen());  
 mt\_vec.push(rng.gen());  
 }  
 }  
 let mr = Matrix::new(size, size, mr\_vec);  
 let mt = Matrix::new(size, size, mt\_vec);  
 p.sort\_by(|a, b| a.partial\_cmp(b).unwrap());  
 let mp = Matrix::new(1, size, p);  
 mp \* (mr \* mt)  
}  
  
  
  
fn main() {  
 let size: usize = 500;  
 let s1 = size.clone();  
 let s2 = size.clone();  
 let s3 = size.clone();  
  
 let handle1 = thread::spawn(move || {  
 println!("Start F1");  
 f1(s1);  
 println!("End F1");  
 });  
  
 let handle2 = thread::spawn(move || {  
 println!("Start F2");  
 f2(s2);  
 println!("End F2");  
 });  
  
 let handle3 = thread::spawn(move || {  
 println!("Start F3");  
 f3(s3);  
 println!("End F3");  
 });  
  
 handle1.join().unwrap();  
 handle2.join().unwrap();  
 handle3.join().unwrap();  
}

**Приклад роботи програми**

****

Start F1

Start F2

Start F3

End F3

End F2

End F1

