

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

Звіт
із лабораторної роботи №6
із дисципліни «Розподілені і хмарні обчислення»

Виконав:
студент групи КМ-01
Скорденко Д. О.

Керівник:
доцент кафедри ПМА
Ліскін В. О.

ЗМІСТ

Вступ	3
1 Основна частина	4
2 Опис програми [Тестовий приклад]	5
Висновки	7
Додаток Код лістинги	8

ВСТУП

Мета: створити програму для виділення областей зображення, та розпаралелити її.

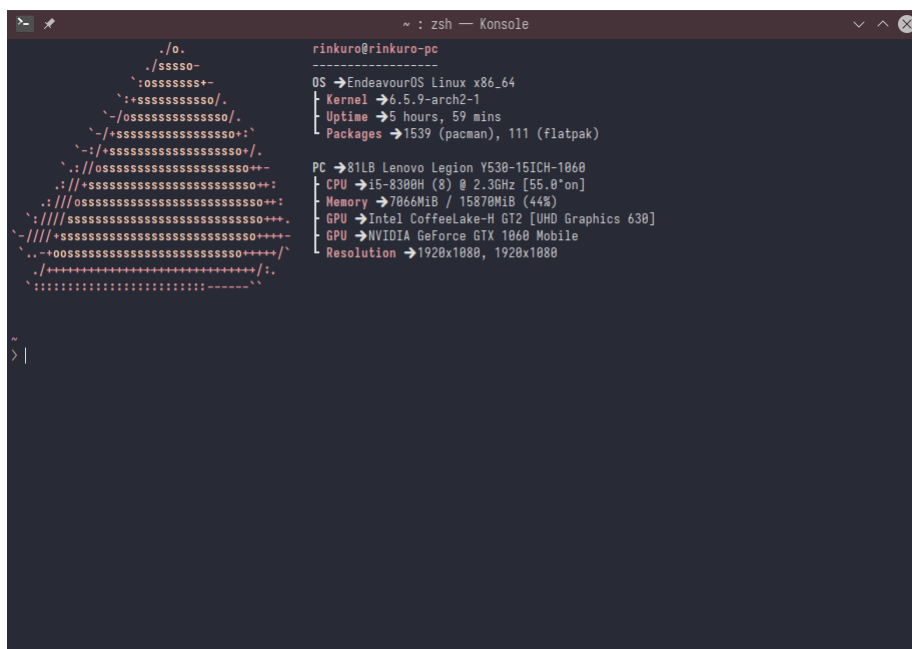
Дослідний приклад:

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

1 ОСНОВНА ЧАСТИНА

Опис програми: Для реалізації паралелізму буде використовуватись 'Rayon'. Для матриць / векторів буде використовуватись 'nalgebra'.

2 ОПИС ПРОГРАМИ [ТЕСТОВИЙ ПРИКЛАД]

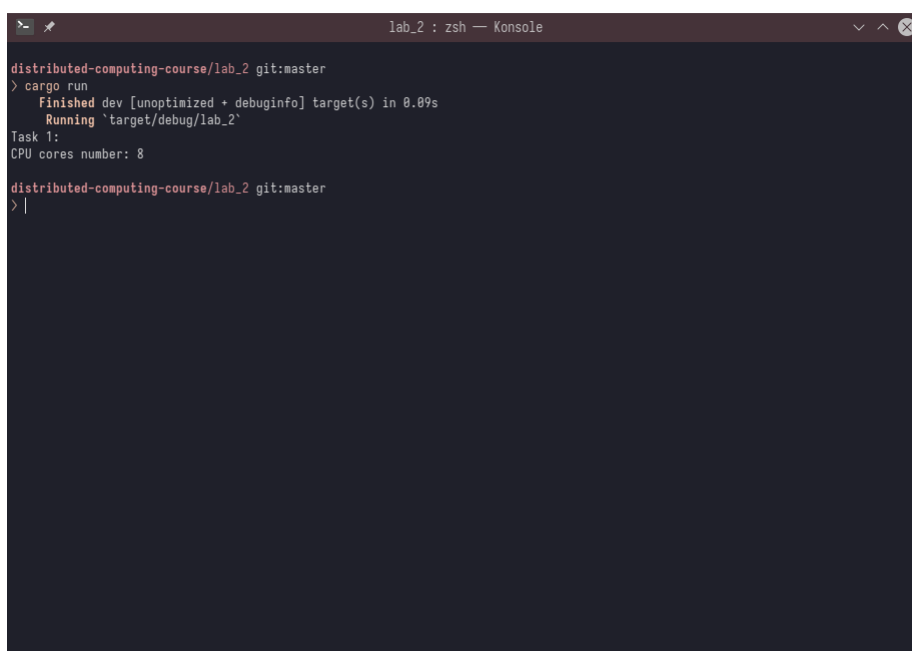


The screenshot shows a terminal window titled "zsh — Konsole" with the user "rinkuro@rinkuro-pc". On the left, there is a large ASCII art logo consisting of various symbols like dots, slashes, and asterisks arranged in a complex pattern. On the right, system information is displayed in a structured format:

```
rinkuro@rinkuro-pc
-----
OS → EndeavourOS Linux x86_64
Kernel → 6.5.9-arch2-1
Uptime → 5 hours, 59 mins
Packages → 1539 (pacman), 111 (flatpak)

PC → 81LB Lenovo Legion Y530-15ICH-1060
CPU → i5-8300H (8) @ 2.3GHz [55.0°C]
Memory → 7066MiB / 15870MiB (44%)
GPU → Intel CoffeeLake-H GT2 [UHD Graphics 630]
GPU → NVIDIA GeForce GTX 1060 Mobile
Resolution → 1920x1080, 1920x1080
```

Рисунок 2.1 – Характеристики системы



The screenshot shows a terminal window titled "lab_2 : zsh — Konsole" with the user "distributed-computing-course/lab_2 git:master". The user has run the command "cargo run", which has finished. The output shows the number of CPU cores:

```
distributed-computing-course/lab_2 git:master
> cargo run
Finished dev [unoptimized + debuginfo] target(s) in 0.09s
Running `target/debug/lab_2`
Task 1:
CPU cores number: 8
distributed-computing-course/lab_2 git:master
>
```

Рисунок 2.2 – К-сть ядер процессора



```
lab_6t : zsh — Konsole
distributed-computing-course/lab_6t git:master
> cargo run
    Finished dev [unoptimized + debuginfo] target(s) in 0.04s
    Running `target/debug/lab_6t`
Original Image:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Labeled Image:

$$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 4 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 3 & 3 & 3 \\ 2 & 0 & 0 & 3 & 0 & 0 \\ 2 & 0 & 0 & 3 & 0 & 6 \end{bmatrix}$$

distributed-computing-course/lab_6t git:master
> |
```

Рисунок 2.3 – Тестовий приклад

ВИСНОВКИ

Було реалізовано виділення областей зображення у паралельному середовищі.

Додаток

Код лістинги

*Примітка: У код лістингах при копіюванні втрачається форматування (не копіюються пробіли). Файли прикріплено до цього pdf (вкладка "прикріплені файли").

Listing 1: main.rs

```
extern crate nalgebra as na;
extern crate rayon;

use std::sync::{Mutex};

use na::DMatrix;
use rayon::prelude::*;

fn label_4_neighbors(image: &DMatrix<u32>) → DMatrix<u32> {
    let (rows, cols) = image.shape();
    let mut labels = DMatrix::from_element(rows, cols, 0u32);
    let _labels = Mutex::new(&mut labels);
    let next_label = Mutex::new(1u32);

    (0..rows).into_par_iter().for_each(|i| {
        (0..cols).into_par_iter().for_each(|j| {
            if image[(i, j)] ≠ 0 {
                let mut neighbor_labels = Vec::new();
```



```

let mut _labels = _labels.lock().unwrap();

if i > 0 && _labels[(i - 1, j)] ≠ 0 {
    neighbor_labels.push(_labels[(i - 1, j)]);
}
if i < rows - 1 && _labels[(i + 1, j)] ≠ 0 {
    neighbor_labels.push(_labels[(i + 1, j)]);
}
if j > 0 && _labels[(i, j - 1)] ≠ 0 {
    neighbor_labels.push(_labels[(i, j - 1)]);
}
if j < cols - 1 && _labels[(i, j + 1)] ≠ 0 {
    neighbor_labels.push(_labels[(i, j + 1)]);
}

match neighbor_labels.iter().cloned().min() {
    Some(min_label) ⇒ {
        _labels[(i, j)] = min_label;
        for &label in &neighbor_labels {
            if label ≠ min_label {
                relabel_regions(&mut _labels, label, min_label);
            }
        }
    }
}

```

```

        None => {
            let mut _next_label = next_label.lock().unwrap();
            _labels[(i, j)] = *_next_label;
            *_next_label += 1;
        }
    }
}

});
});

labels
}

fn relabel_regions(labels: &mut DMatrix<u32>, old_label: u32, new_label: u32) {
    labels
        .iter_mut()
        .for_each(|label| {
            if old_label == *label {
                *label = new_label
            }
        });
}

fn main() {
    let data = DMatrix::from_row_slice(

```

```

        6,
        6,
        &[
            1, 1, 0, 0, 0, 0,
            1, 1, 0, 1, 0, 0,
            1, 0, 0, 0, 0, 0,
            0, 0, 1, 1, 1, 1,
            1, 0, 0, 1, 0, 0,
            1, 0, 0, 1, 0, 1
        ],
    );

    let labeled_image = label_4_neighbors(&data);

    println!("Original Image:");
    println!("{}", data);
    println!("Labeled Image:");
    println!("{}", labeled_image);
}

```