НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №*1*

із дисципліни «Розподілені і хмарні обчислення»

|  |  |
| --- | --- |
| Виконав: | Керівник: |
| студент групи КМ-03 | Ліскін В. О*.* |
| Шаповалов Г. Г. |  |

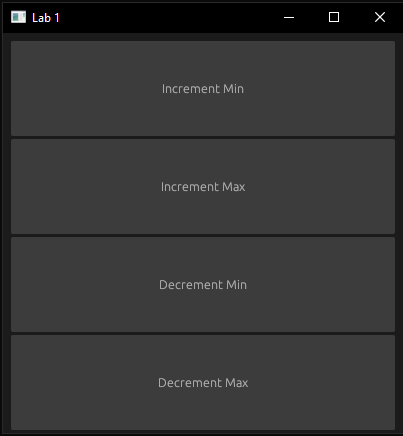
Київ — 2023

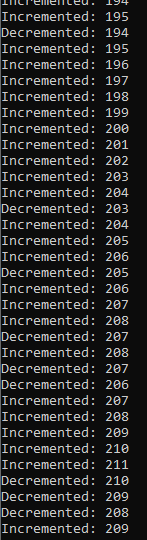
Мета роботи

Навчитись працювати з потоками

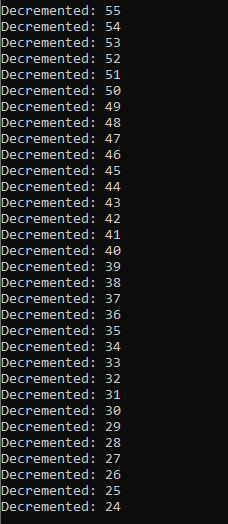
Опис програми

Створено інтерфейс із кнопками для регулювання пріоритетності потоків на операційній системі віндовс.





До натискання Decrement Max



Після

Лістинг програми

#[macro\_use]

extern crate lazy\_static;

use eframe::{egui, epi};

use std::sync::{Arc, Mutex};

use std::thread;

use std::time::Duration;

use thread\_priority::\*;

use std::os::windows::io::AsRawHandle;

use winapi::ctypes::c\_void as winapi\_c\_void;

lazy\_static! {

    static ref PRIORITY\_MIN: ThreadPriority = ThreadPriority::Os(WinAPIThreadPriority::Lowest.into());

    static ref PRIORITY\_MAX: ThreadPriority = ThreadPriority::Os(WinAPIThreadPriority::Highest.into());

}

fn increment(refr: Arc<Mutex<i32>>) {

    loop {

        let mut num = refr.lock().unwrap();

        \*num += 1;

        println!("Incremented: {}", num);

        thread::sleep(Duration::from\_millis(100));

    }

}

fn decrement(refr: Arc<Mutex<i32>>) {

    loop {

        let mut num = refr.lock().unwrap();

        \*num -= 1;

        println!("Decremented: {}", num);

        thread::sleep(Duration::from\_millis(100));

    }

}

struct App {

    inc\_handle: \*mut winapi\_c\_void,

    dec\_handle: \*mut winapi\_c\_void,

}

impl epi::App for App {

    fn name(&self) -> &str {

        "Lab 1"

    }

    fn update(&mut self, ctx: &egui::CtxRef, \_frame: &mut epi::Frame<'\_>) {

        egui::CentralPanel::default().show(ctx, |ui| {

            let button\_size = egui::vec2(ui.available\_width(), 95.0);

            if ui.add\_sized(button\_size, egui::Button::new("Increment Min")).clicked() {

                set\_thread\_priority(self.inc\_handle, \*PRIORITY\_MIN).expect("Failed to set thread priority");

            }

            if ui.add\_sized(button\_size, egui::Button::new("Increment Max")).clicked() {

                set\_thread\_priority(self.inc\_handle, \*PRIORITY\_MAX).expect("Failed to set thread priority");

            }

            if ui.add\_sized(button\_size, egui::Button::new("Decrement Min")).clicked() {

                set\_thread\_priority(self.dec\_handle, \*PRIORITY\_MIN).expect("Failed to set thread priority");

            }

            if ui.add\_sized(button\_size, egui::Button::new("Decrement Max")).clicked() {

                set\_thread\_priority(self.dec\_handle, \*PRIORITY\_MAX).expect("Failed to set thread priority");

            }

        });

    }

}

fn main() {

    let counter = Arc::new(Mutex::new(0));

    let counter\_clone\_inc = Arc::clone(&counter);

    let inc\_thread = thread::spawn(move || {

        increment(counter\_clone\_inc);

    });

    let counter\_clone\_dec = Arc::clone(&counter);

    let dec\_thread = thread::spawn(move || {

        decrement(counter\_clone\_dec);

    });

    let inc\_handle = inc\_thread.as\_raw\_handle() as \*mut winapi\_c\_void;

    let dec\_handle = dec\_thread.as\_raw\_handle() as \*mut winapi\_c\_void;

    let app = App {

        inc\_handle,

        dec\_handle,

    };

    let mut native\_options = eframe::NativeOptions::default();

    native\_options.initial\_window\_size = Some(egui::vec2(400.0, 400.0));

    eframe::run\_native(Box::new(app), native\_options);

}