**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**

**Кафедра систем штучного інтелекту**



# **Лабораторна робота**

з дисципліни

« Технології розподілених систем та паралельних обчислень»

**Виконав:**

студент групи КН-314

Ляшеник О. А

**Викладач:**

Мочурад Л. І.

**Лабораторна робота №2**

**Тема:** Розподіл роботи між потоками засобами OpenMP

**Мета:** Вивчити розподіл роботи між потоками засобами OpenMP і навчитися їх застосовувати.

**Хід роботи**

**Функціональний код ( Python ) :**

import random  
import threading  
  
def fill\_matrix(matrix, size):  
 *"""Fills a matrix with random integers between -10 and 10."""*  
for i in range(size):  
 for j in range(size):  
 matrix[i][j] = random.randint(-10, 10)  
  
def print\_matrix(matrix, size):  
 *"""Prints the matrix."""*  
for i in range(size):  
 for j in range(size):  
 print(matrix[i][j], end=" ")  
 print()  
  
def find\_max\_elements(matrix, result, row, schedule\_type, chunk\_size):  
 *"""Calculates the sum of positive elements in a given row of the matrix."""*  
sum\_positive = max(item for item in matrix[row])  
 result[row] = sum\_positive  
 thread\_id = threading.get\_ident()  
 print(f"Потік {thread\_id} обробив рядок {row} з розподілом {schedule\_type}, чанк = {chunk\_size}")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 size = int(input("Введіть розмір матриці (n x n): "))  
 matrix\_a = [[0] \* size for \_ in range(size)]  
 matrix\_b = [[0] \* size for \_ in range(size)]  
 sum\_a = [0] \* size  
 sum\_b = [0] \* size  
  
 # Filling matrices A and B with random values  
 fill\_matrix(matrix\_a, size)  
 fill\_matrix(matrix\_b, size)  
  
 # Printing matrices A and B  
 print("Матриця A:")  
 print\_matrix(matrix\_a, size)  
 print("\nМатриця B:")  
 print\_matrix(matrix\_b, size)  
  
 # Processing matrix A in threads  
 threads = []  
 for i in range(size):  
 thread = threading.Thread(target=find\_max\_elements, args=(matrix\_a, sum\_a, i, "dynamic", 6))  
 thread.start()  
 threads.append(thread)  
 for thread in threads:  
 thread.join()  
  
 threads.clear()  
 # Processing matrix B in threads  
 for i in range(size):  
 thread = threading.Thread(target=find\_max\_elements, args=(matrix\_b, sum\_b, i, "guided", 6))  
 thread.start()  
 threads.append(thread)  
 for thread in threads:  
 thread.join()  
  
 # Printing the sums of positive elements in matrices A and B  
 print("\nСума позитивних елементів у рядках матриці A:")  
 for sum\_pos in sum\_a:  
 print(sum\_pos, end=" ")  
  
 print("\n\nСума позитивних елементів у рядках матриці B:")  
 for sum\_pos in sum\_b:  
 print(sum\_pos, end=" ")  
 print()

**Вивід програми:**

Введіть розмір матриці (n x n): 5

Матриця A:

-8 -2 2 9 -4

1 1 -2 -8 -5

-6 -10 8 -2 -5

9 -2 7 6 4

-5 8 0 -2 2

Матриця B:

10 -8 -2 3 2

10 -9 -1 -6 4

6 -6 10 -8 10

10 -4 3 -10 3

-1 -6 -5 -8 7

Потік 1924 обробив рядок 0 з розподілом dynamic, чанк = 6

Потік 4880 обробив рядок 1 з розподілом dynamic, чанк = 6

Потік 8600 обробив рядок 2 з розподілом dynamic, чанк = 6

Потік 15600 обробив рядок 3 з розподілом dynamic, чанк = 6

Потік 14760 обробив рядок 4 з розподілом dynamic, чанк = 6

Потік 15640 обробив рядок 0 з розподілом guided, чанк = 6

Потік 2808 обробив рядок 1 з розподілом guided, чанк = 6

Потік 12284 обробив рядок 2 з розподілом guided, чанк = 6

Потік 2936 обробив рядок 3 з розподілом guided, чанк = 6

Потік 7692 обробив рядок 4 з розподілом guided, чанк = 6

Сума позитивних елементів у рядках матриці A:

9 1 8 9 8

Сума позитивних елементів у рядках матриці B:

10 10 10 10 7

**Висновок:**

У ході виконання цієї лабораторної роботи було досліджено та практично застосовано різні аспекти роботи з матрицями в Python, включаючи їх створення, заповнення випадковими значеннями, обчислення максимальних позитивних елементів у кожному рядку, та використання багатопотоковості для оптимізації обчислень. Було продемонстровано, як ефективно застосовувати багатопоточність для паралельної обробки даних у матрицях, що може суттєво покращити продуктивність обчислень у задачах з великим обсягом даних. Також робота показала важливість чистого коду та добре структурованих програм, які легко читаються та підтримуються.