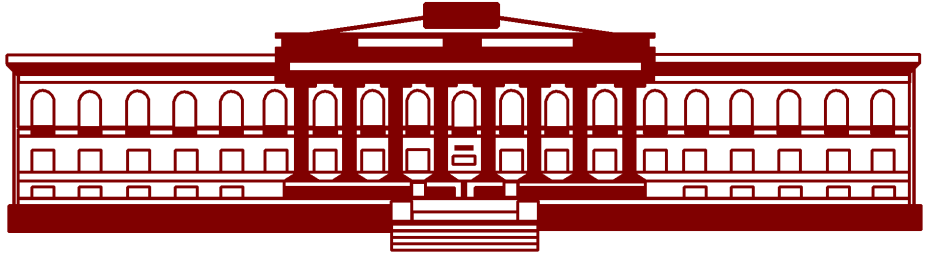
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**імені ТАРАСА ШЕВЧЕНКА**

Факультет інформаційних технологій

**Кафедра прикладних інформаційних систем**

****

**Звіт**

до лабораторної роботи №3

з дисципліни «**Паралельні та розподілені обчислення**»

ВАРІАНТ №16

**Виконано:**

студ. групи ПП- 31, підгрупа 2

Селецький Віктор

**Київ – 2023**

Тема: Паралельне програмування за допомогою засобів OpenMP

Ціль: освоїти реалізацію паралельних обчислень за допомогою

засобів OpenMP та виконати порівняльне оцінювання часу виконання

програм при різній реалізації з та без використання OpenMP та програм,

реалізованих як результат лабораторної роботи No1.

Завдання: На базі однієї із операційних систем (за вибором студента) на одній із

мов програмування, що мають можливість підключення відповідних засобів

OpenMP (С / С++ / Fortran / Python / Java / тощо) створити програмний

застосунок, який підраховує суму перших N членів заданої послідовності

(формулу для підрахунку i-го елемента взяти з таблиці 1).

Програма повинна виконувати такі дії:

- запитувати значення змінної N

- у циклі від 0 до N розраховувати значення i-го елемента послідовності

ai. Додавати це значення до суми. Виводити результат суми та

витраченого на розрахунок часу на екран.

Підібрати таке значення N, щоб програма виконувалася кілька десятків

секунд (50-500 с). Для цього почати з маленьких значень, поступово

збільшуючи їх.

У звіті навести код програми, результати її виконання: суму та час

виконання за кількох значень N. Порівняти результати при різній кількості

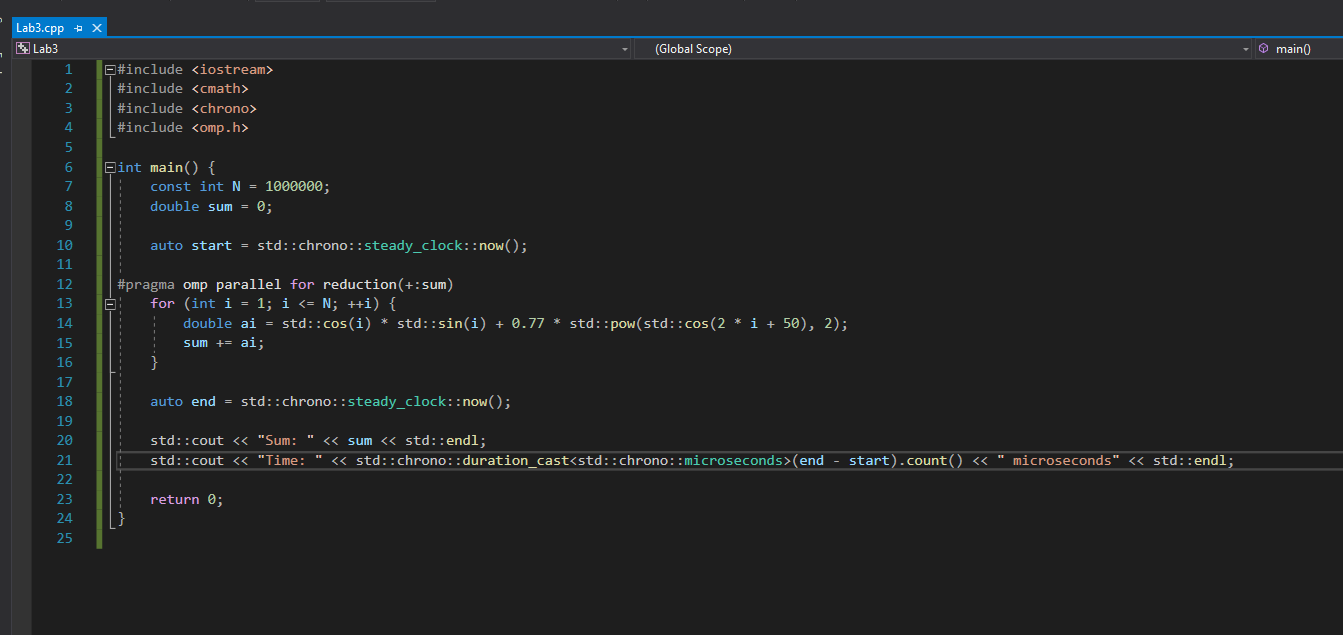
потоків.

Порівняти результати з результатами практичної роботи No1 та No2.

Зробити висновок.



Виконання:



Код:

#include <iostream>

#include <cmath>

#include <chrono>

#include <omp.h>

int main() {

const int N = 1000000;

double sum = 0;

auto start = std::chrono::steady\_clock::now();

#pragma omp parallel for reduction(+:sum)

for (int i = 1; i <= N; ++i) {

double ai = std::cos(i) \* std::sin(i) + 0.77 \* std::pow(std::cos(2 \* i + 50), 2);

sum += ai;

}

auto end = std::chrono::steady\_clock::now();

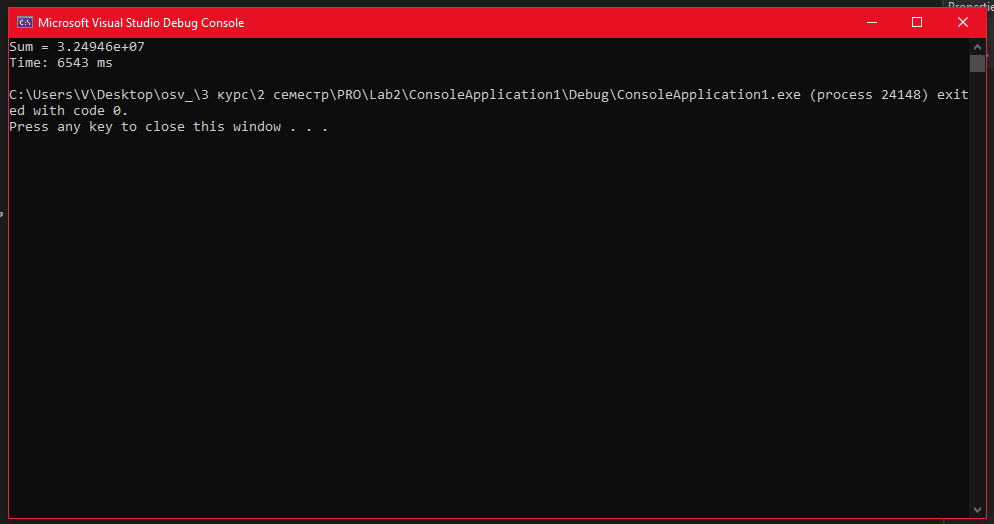
std::cout << "Sum: " << sum << std::endl;

std::cout << "Time: " << std::chrono::duration\_cast<std::chrono::microseconds>(end - start).count() << " microseconds" << std::endl;

return 0;

}

Результат виконання:



(100000000 членів прогресії)

Таблиця для порівняння:

| **Кількість членів** | **C++ (без мультипоточності)** | **C++ (з мультипоточністю)** | **Python (без мультипоточності)** | **Python (з мультипоточністю)** | **C++ (OpenMP)** |
| --- | --- | --- | --- | --- | --- |
| 100 | 0.000008 сек | 0.000033 сек | 0.001820 сек | 0.000340 сек | 0.000298 сек |
| 1000 | 0.000033 сек | 0.000048 сек | 0.000042 сек | 0.000076 сек | 0.001325 сек |
| 10000 | 0.000310 сек | 0.000218 сек | 0.000571 сек | 0.000295 сек | 0.001058 сек |
| 999999 | 0.025255 сек | 0.004491 сек | 0.056196 сек | 0.013505 сек | 0.076945 сек |