#include <iostream>

#include <omp.h>

#include <chrono>

#include <cmath>

#include <iomanip>

using namespace std;

using namespace std::chrono;

// Versión secuencial del cálculo de PI

double calcularPiSecuencial(long n) {

    double pi = 0.0;

    for (long i = 0; i < n; i++) {

        double term = 1.0 / (2 \* i + 1);

        if (i % 2 == 0) {

            pi += term;

        } else {

            pi -= term;

        }

    }

    return pi \* 4;

}

// Versión paralela del cálculo de PI usando OpenMP

double calcularPiParalelo(long n) {

    double pi = 0.0;

    #pragma omp parallel for reduction(+:pi)

    for (long i = 0; i < n; i++) {

        double term = 1.0 / (2 \* i + 1);

        if (i % 2 == 0) {

            pi += term;

        } else {

            pi -= term;

        }

    }

    return pi \* 4;

}

void ejecutarPrueba(long n) {

    cout << "Calculando PI con n = " << n << endl;

    // Versión secuencial

    auto inicio\_sec = high\_resolution\_clock::now();

    double pi\_sec = calcularPiSecuencial(n);

    auto fin\_sec = high\_resolution\_clock::now();

    auto tiempo\_sec = duration\_cast<microseconds>(fin\_sec - inicio\_sec).count();

    // Versión paralela

    auto inicio\_par = high\_resolution\_clock::now();

    double pi\_par = calcularPiParalelo(n);

    auto fin\_par = high\_resolution\_clock::now();

    auto tiempo\_par = duration\_cast<microseconds>(fin\_par - inicio\_par).count();

    // Calcular diferencia de tiempo y aceleración

    auto diferencia\_tiempo = tiempo\_sec - tiempo\_par;

    double aceleracion = (tiempo\_sec > 0) ? static\_cast<double>(tiempo\_sec)/tiempo\_par : 0;

    // Mostrar resultados

    cout << fixed << setprecision(15);

    cout << "Secuencial: PI ≈ " << pi\_sec << " - Tiempo: " << tiempo\_sec << " μs" << endl;

    cout << "Paralelo:   PI ≈ " << pi\_par << " - Tiempo: " << tiempo\_par << " μs" << endl;

    cout << "Diferencia de tiempo: " << diferencia\_tiempo << " μs (secuencial - paralelo)" << endl;

    cout << "Aceleración: " << aceleracion << "x" << endl;

    cout << "Diferencia entre resultados: " << abs(pi\_sec - pi\_par) << endl << endl;

}

int main() {

    // Configurar número de hilos (opcional)

    // omp\_set\_num\_threads(4);

    // Ejecutar pruebas para los diferentes valores de n

    ejecutarPrueba(1000);

    ejecutarPrueba(10000);

    ejecutarPrueba(3);

    return 0;

}