МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

КАФЕДРА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ



**ЗВІТ**

про виконання лабораторної роботи №3

з дисципліни: «Технології розподілених систем та паралельних обчислень»

на тему:

«Завантаження та синхронізація в OpenMP»

Виконав:

студент групи КН-308

*Матвіїв М.А.*

Прийняла:

*Мочурад Л.І.*

Львів 2020

**Варіант 13**

**Мета роботи:** Вивчити роботу з операторами керування паралельними потоками засобами OpenMP

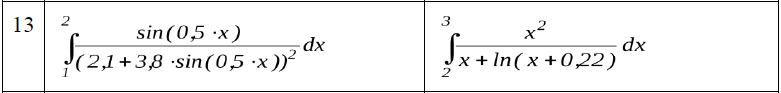
**Завдання:**

Розробити програму паралельного розрахунку означеного інтегралу для функції з кроком дискретизації 0.00002, використовуючи вхідні дані відповідно до завдання. Провести чисельні експерименти розрахунку визначеного інтегралу з різним кроком дискретизації і різною кількістю ядер процесора. Результати оформити у вигляді таблиці. Побудувати графік залежності часу обчислення від кількості ядер та відповідного кроку дискретизації.

Результати чисельних експериментів вивести у текстовий файл. При цьому перед виводом кожен потік повинен виставити блокування за допомогою механізму замків. У текстовий файл всі потоки по черзі повинні вивести повідомлення "The beginning of the closed section..." і " The end of the closed section...". Якщо при цьому між двома повідомленнями від одного потоку зустрінуться повідомлення від інших потоків про невдалу спробу увійти до закритої секції, вони також повинні бути записані у файл.

У паралельній області за допомогою директиви single або master вивести наступні дані: номер лабораторної роботи; назва лабораторної роботи; групу студента; ПІБ студента; номер варіанту; завдання.

Студенти, чий номер у журналі викладача при діленні на 4 дає остачу 3 – обчислюють наближене значення інтегралу за формулою лівих прямокутників; остачу 2 – за формулою правих прямокутників, остачу 1 – за формулою середніх прямокутників і без остачі – за формулою трапеції.



**Код програми:**

#include <iostream>

#include <omp.h>

#include <fstream>

using namespace std;

omp\_lock\_t lock;

//ofstream file("D:\\University\\Paralel\\text1.txt");

ofstream file("D:\\University\\Paralel\\text2.txt");

double f(double x) {

//return sin(0.5\*x)/pow((2.1+3.8\*sin(0.5\*x)),2);

return pow(x,2)/(x+log(x+0.22));

}

void single\_result(string s) {

#pragma omp master

{

file << "Lab № 3" << endl;

file << "Download and synchronization in OpenMP" << endl;

file << "KN - 308" << endl;

file << "Matviiv Mykola" << endl;

file << "Variant 13 " << endl;

file << s << endl;

}

}

double ser\_rectangle(double a, double b, double n, int thread\_num) {

double count = (b - a) / n;

double sum = 0.0, fx;

double start, finsh;

int i, j;

omp\_init\_lock(&lock);

//single\_result("sin(0.5\*x)/pow((2.1+3.8\*sin(0.5\*x)),2)");

single\_result("pow(x,2)/(x+log(x+0.22))");

omp\_set\_num\_threads(thread\_num);

start = omp\_get\_wtime();

#pragma omp parallel

{

#pragma omp parallel for private(i, fx)

for (i = 0; i < count; i++) {

fx = f((a+b)/2 + i \* n);

#pragma omp atomic

sum += fx;

}

}

double res = sum \* n;

#pragma omp parallel

{

while (!omp\_test\_lock(&lock)) {

#pragma omp critical

file << "Wait.. thread " << omp\_get\_thread\_num() << endl;

}

#pragma omp critical

file << "The beginning of closed section" << endl;

#pragma omp critical

file << "Thread: " << omp\_get\_thread\_num() << endl;

#pragma omp critical

file << "Thread " << omp\_get\_thread\_num() << " result: " << res << endl;

#pragma omp critical

file << "The end of closed section" << endl << endl;

omp\_unset\_lock(&lock);

}

omp\_destroy\_lock(&lock);

finsh = omp\_get\_wtime();

cout << "Time for " << thread\_num << " threads is: " << finsh - start << endl;

return sum \* n;

}

int main() {

double n;

double a, b;

int num;

cout << "Input a: ";

cin >> a;

cout << "Input b: ";

cin >> b;

cout << "Input step: ";

cin >> n;

cout << "Input thread num: ";

cin >> num;

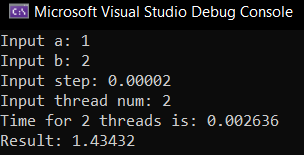
cout << "Result: " << ser\_rectangle(a, b, n, num) << endl;

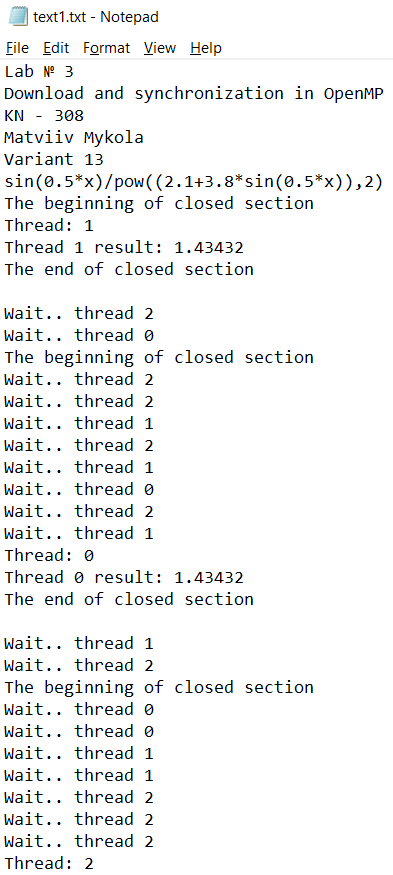
return 0;

}

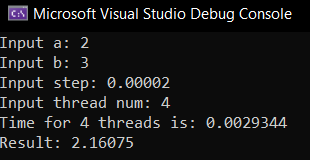
**Результати роботи:**

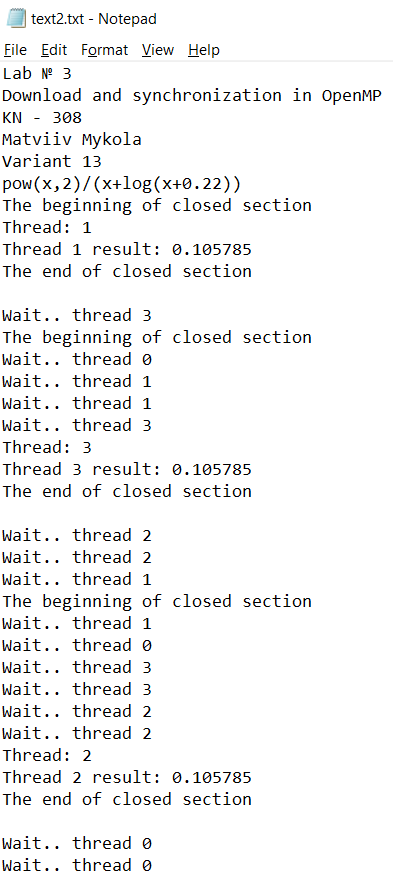
Integral 1





Integral 2

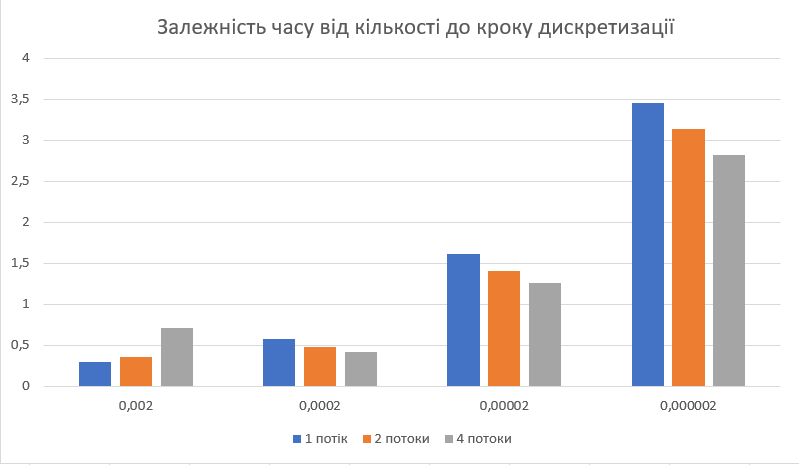




**Integral №1**

|  |  |  |  |
| --- | --- | --- | --- |
| Крок дискретизації | Час, с | | |
| 1 потік | 2 потоки | 4 потоки |
| 0.002 | **0.295174** | **0.351637** | **0.712846** |
| 0.0002 | **0,580391** | **0.** **478106** | **0.4154516** |
| 0.00002 | **1.61483** | **1.43432** | **1.264138** |
| 0.000002 | **3.459882** | **3.138761** | **2.816846** |

*Таблиця 1. Залежність часу від кількості до кроку дискретизації*

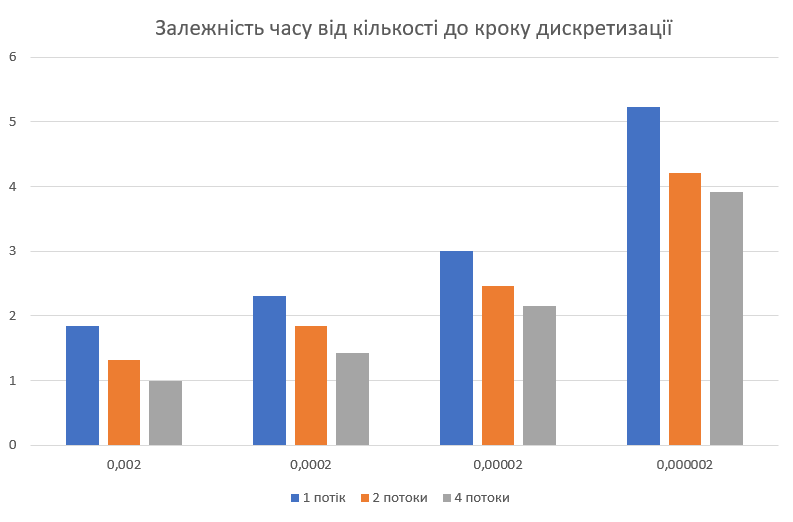
****

*Графік 1. Залежність часу від кількості до кроку дискретизації*

**Integral №2**

|  |  |  |  |
| --- | --- | --- | --- |
| Крок дискретизації | Час, с | | |
| 1 потік | 2 потоки | 4 потоки |
| 0.002 | **1.84161** | **1.316516** | **0.994161** |
| 0.0002 | **2.31435** | **1.846516** | **1.41981** |
| 0.00002 | **2.99851** | **2.468161** | **2.16075** |
| 0.000002 | **5.234856** | **4.216513** | **3.91561** |

*Таблиця 2. Залежність часу від кількості до кроку дискретизації*

****

*Графік 2. Залежність часу від кількості до кроку дискретизації*

**Висновок**

Як ми можемо побачити з графіків, що чим більший в нас крок дискретизації, тоді можна застосовувати багато поточність. Це полягає в тому що застосовується блокування потоків, а чим менший крок, тим менше ітерацій в нас виконується.