**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНОМУ УНІВЕРСИТЕТІ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”**



**Лабораторна робота**

з дисципліни

«Технології розподілених систем та паралельних обчислень»

**Виконав:**

студент групи КН-308

Гецянин Дмитро

**Викладач:**

Мочурад Л.І.

Львів – 2020р.

**Лабораторна робота №3**

**Тема: Завантаження та синхронізація в OpenMP**

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

**Варіант №3**

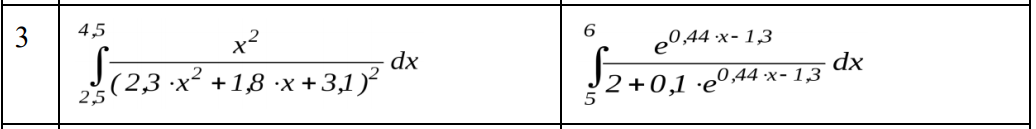
**Завдання.**

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.



**Програмний код**

#include <iostream>

#include <iomanip>

#include <fstream>

#include <windows.h>

#include <omp.h>

**using** **namespace** std;

**double** **f**(**double** x)

{

**return** (pow(x,**2**) / pow(**2.3**\*x + **1.8**\*x + **3.1**,**2**));

}

/\*double f(double x)

{

double param = 0.44\*x - 1.3;

return (exp(param) / (2 + 0.1 \* exp(param)));

}\*/

**omp\_lock\_t** lock;

**int** **main**()

{

ofstream myfile("test.txt");

**double** a, b;

cout << "Input a = ";

cin >> a;

cout << "**\n**Input b = ";

cin >> b;

**int** n, precision;

cout << "**\n**Input n = ";

cin >> n;

cout << "**\n**Input precision = ";

cin >> precision;

**double** s = (f(a) + f(b)) / **2**;

**double** h = (b - a) / n;

**double** sum;

omp\_init\_lock(&lock);

**int** i, thread\_num;

**clock\_t** time = clock();

omp\_set\_num\_threads(**2**);

#pragma omp parallel private(i, thread\_num) shared(h, n, s, sum)

{

#pragma omp master

{

cout << "**\n**Lab 3: Synchronization in OpenMP" << endl;

cout << "Hetsyanin Dmytro KH-308" << endl;

cout << "Variant 3" << endl;

}

**for** (i = **1**; i <= n - **1**; i++)

{

sum = f(a + i \* h);

thread\_num = omp\_get\_thread\_num();

omp\_set\_lock(&lock);

myfile << "The beginning of the closed section, thread " << thread\_num << endl;

s += sum;

myfile << s << "**\n**";

myfile << "The end of the closed section, thread " << thread\_num << endl;

omp\_unset\_lock(&lock);

}

}

omp\_destroy\_lock(&lock);

**double** I = h \* s;

cout << "**\n**Result = " << setprecision(precision) << I << endl;

cout << "**\n**Time of execution = " << ((**float**)(clock() - time)) / CLK\_TCK << " seconds**\n**";

**return** **0**;

}

**Результати виконання програмного коду**

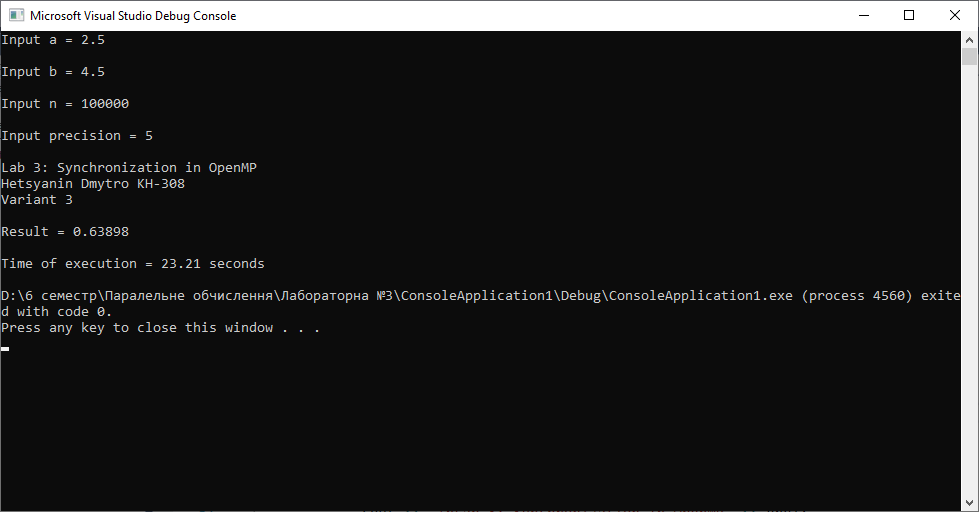


Рис. 1. Вивід в консоль

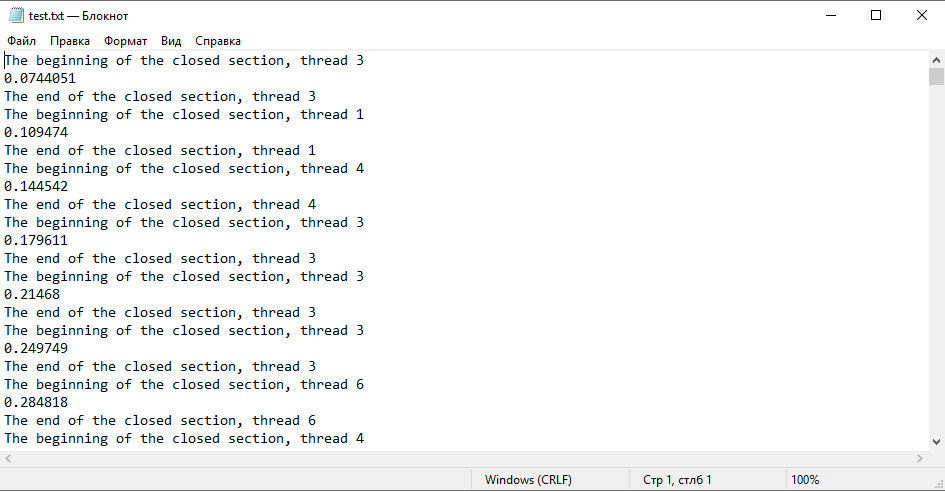
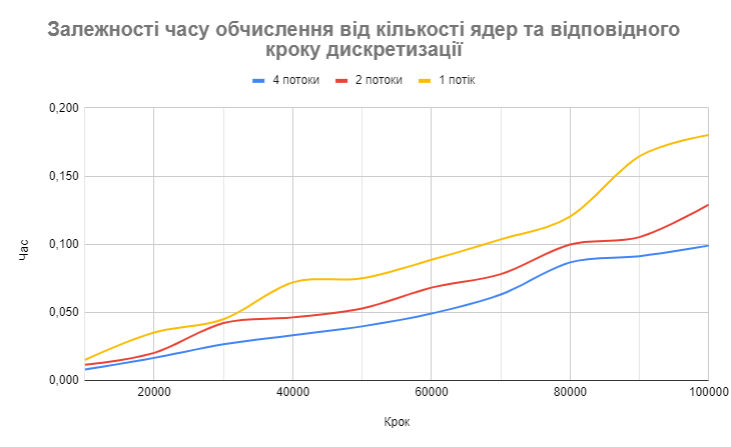


Рис. 2. Вивід в текстовий файл

|  |  |  |  |
| --- | --- | --- | --- |
| **Крок, n** | **Час виконання, с** | | |
| **1 потік** | **2 потоки** | **4 потоки** |
| 10000 | 1,517 | 1,151 | 0,799 |
| 20000 | 3,522 | 2,025 | 1,657 |
| 30000 | 4,491 | 4,214 | 2,654 |
| 40000 | 7,198 | 4,630 | 3,318 |
| 50000 | 7,495 | 5,282 | 3,970 |
| 60000 | 8,854 | 6,812 | 4,912 |
| 70000 | 10,357 | 7,805 | 6,305 |
| 80000 | 12,029 | 9,979 | 8,667 |
| 90000 | 16,458 | 10,523 | 9,123 |
| 100000 | 18,012 | 12,898 | 9,898 |

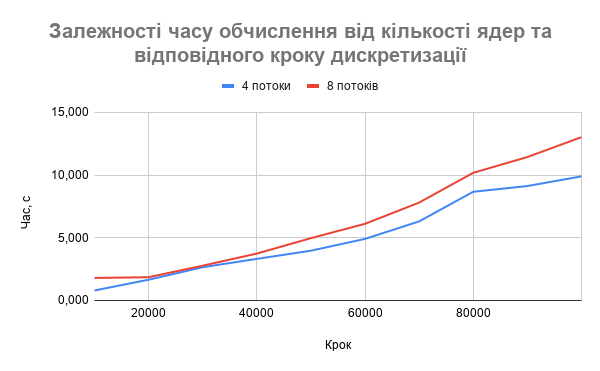
Таблиця 1. Результати при різному кроці дискретизації



Графік 1. Залежність часу обчислення від кількості ядер та відповідного кроку дискретизації

|  |  |  |
| --- | --- | --- |
| **Крок, n** | **Час виконання, с** | |
| **8 потоків** | **4 потоки** |
| 10000 | 1,799 | 0,799 |
| 20000 | 1,857 | 1,657 |
| 30000 | 2,777 | 2,654 |
| 40000 | 3,741 | 3,318 |
| 50000 | 4,970 | 3,970 |
| 60000 | 6,112 | 4,912 |
| 70000 | 7,805 | 6,305 |
| 80000 | 10,179 | 8,667 |
| 90000 | 11,435 | 9,123 |
| 100000 | 13,020 | 9,898 |

Таблиця 2. Результати при різному кроці дискретизації



Графік 2. Залежність часу обчислення від кількості ядер та відповідного кроку дискретизації

**Висновок.** Виконуючи лабораторну роботу, я навчився виконувати роботу з операторами керування паралельними потоками засобами OpenMP. Дослідив залежність часу обчислення від кількості потоків та відповідного кроку дискретизації.