МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра систем штучного інтелекту

ЗВІТ

до лабораторної роботи №3

з дисципліни: **“Теорія розподілених систем**

**та паралельних обчислень”**

на тему:

**«Завантаження та синхронізація OpenMP»**

**Виконав:**

студент групи KH-318

Гавриляк Віталій

**Прийняла:**

асистент Мочурад Л. І.

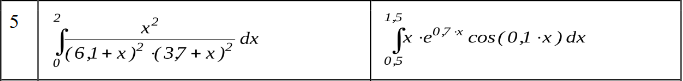
Львів-2019

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

5 варіант

**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.



**Код:**

#include <omp.h>

#include <stdio.h>

#include <fstream>

#include <cmath>

#include <iostream>

using namespace std;

int main() {

int i, j, n;

double a, b, p=0.0002, x, s, h, z;

double t1, t2, t[5];

ofstream fout("text\_lab3.txt");

a=0; b=2;

//a=0.5; b=1.5;

//system("pause");

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

t1 = omp\_get\_wtime();

h=p;

n=(b-a)/h;

s=(a\*a/pow(6.1+a,2)\*pow(3.7+a,2)+a\*a/pow(6.1+b,2)\*pow(3.7+b,2)/2);

//s=(a\*exp(0.7\*a)\*cos(0.1\*a)+b\*exp(0.7\*b)\*cos(0.1\*b)/2);

for (i=1; i<=n; i++){

x=a+h\*i;

z=x\*x/pow(6.1+x,2)\*pow(3.7+x,2);

//z=x\*exp(0.7\*x)\*cos(0.1\*x);

s=s+z;

}

s=s\*h;

cout<<s<<endl;

t2 = omp\_get\_wtime();

t[0]=t2-t1;

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

for (int k=1; k<=3; k++){

int kk=pow(2, k-1);

h=p;

n=(b-a)/h;

s=(a\*a/pow(6.1+a,2)\*pow(3.7+a,2)+a\*a/pow(6.1+b,2)\*pow(3.7+b,2)/2);

//s=(a\*exp(0.7\*a)\*cos(0.1\*a)+b\*exp(0.7\*b)\*cos(0.1\*b)/2);

t1 = omp\_get\_wtime();

#pragma omp parallel for num\_threads(kk) private(x, z)

for (i=1; i<=n; i++){

x=a+h\*i;

z=x\*x/pow(6.1+x,2)\*pow(3.7+x,2);

//z=x\*exp(0.7\*x)\*cos(0.1\*x);

#pragma omp atomic

s=s+z;

}

s=s\*h;

t2 = omp\_get\_wtime();

cout<<s<<endl;

t[k]=t2-t1;

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

omp\_lock\_t lock;

omp\_init\_lock(&lock);

#pragma omp parallel

{

omp\_set\_lock(&lock);

fout<<"The beginning of the closed section (thread "<<omp\_get\_thread\_num()<<")"<<endl;

fout<<"The end of the closed section (thread "<<omp\_get\_thread\_num()<<")"<<endl;

omp\_unset\_lock(&lock);

}

omp\_destroy\_lock(&lock);

#pragma omp parallel

{

#pragma omp master

{

fout<<"\nLab. robota #3"<<endl;

//four<<<<endl;

fout<<"KN-318"<<endl;

fout<<"Gavryliak V.T."<<endl;

fout<<"Variant 5"<<endl;

}

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

cout<<"\nt0 = "<<t[0]<<endl;

cout<<"\nt1 = "<<t[1]<<"\np1 = "<<t[0]/t[1]<<"\ne1 = "<<t[0]/t[1]/1<<endl;

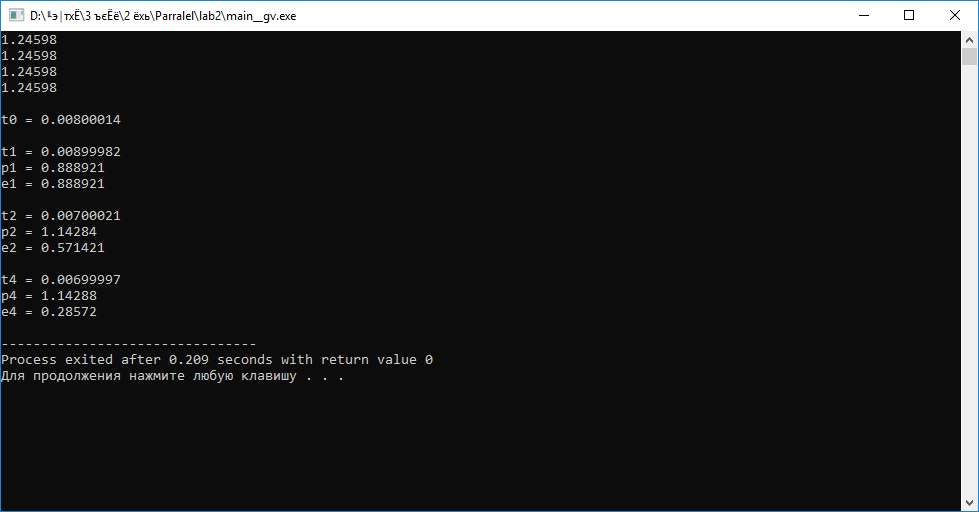
cout<<"\nt2 = "<<t[2]<<"\np2 = "<<t[0]/t[2]<<"\ne2 = "<<t[0]/t[2]/2<<endl;

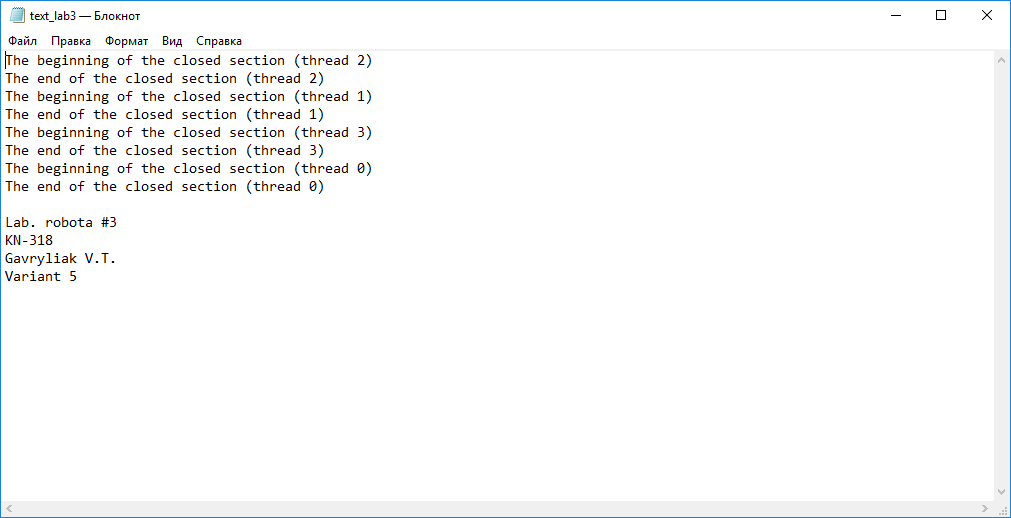
cout<<"\nt4 = "<<t[3]<<"\np4 = "<<t[0]/t[3]<<"\ne4 = "<<t[0]/t[3]/4<<endl;

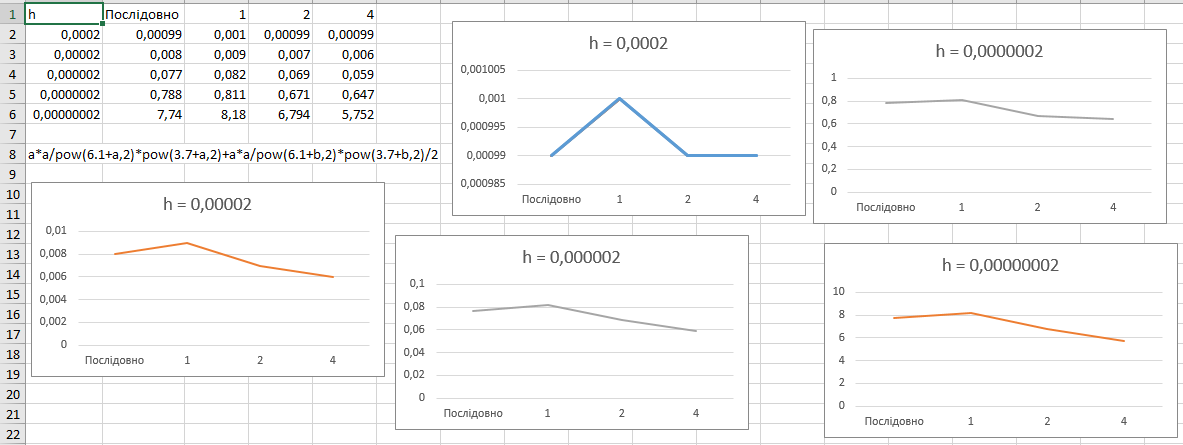
// cout<<"\nt8 = "<<t[4]<<"\np8 = "<<t[0]/t[4]<<"\ne8 = "<<t[0]/t[4]/8<<endl;

}

**Чисельні експеременти:**







**Висновок:** в ході цієї роботи я вивчив роботу з операторами керування паралельними потоками засобами OpenMP.