МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА  
Факультет електроніки і комп’ютерних технологій  
Кафедра системного проектування

**Звіт про виконання лабораторної роботи №3**  
з начальної дисципліни

«Паралельне програмування»

на тему:

**«Синхронізація в OpenMP програмах»**

**Виконав:**  
студент групи ФЕП-22

Линва В. А.

**Львів – 2021**

**Хід роботи**

1. Написав програму та запустив її згідно індивідуального завдання
2. Використав синхронізацію потоків на основі виразів явної синхронізації та технології замків.
3. Реалізував вибір методу синхронізації в самій програмі.

Індивідуальне завдання номер 3 – «Написати програму, яка наближено обчислює визначений інтеграл за формулою трапеції»

Код програми:

#include <iostream>

#include <math.h>

#include <omp.h>

using namespace std;

//Variant - 3

double f(double x)

{

return sin(x);

}

int main(int t)

{

cout << " Write 1 for choose cale - Synchronization with directives\n Write 2 for choose case - Synchronyzation with locks\n Your choose: ";

cin >> t;

switch (t)

{

case 1:

#pragma omp parallel private (Integral, n) shared(a,b,h)

{

double start\_time, end\_time;

double Integral;

double a, b; // Задаю відрізок інтегрування a=0.0 b=1.0

double h; // Задаю крок інтегрування 0.1

double n; // Число розбивань

cout << "Enter the beginning of the integration segment: "; // Початок відрізку

cin >> a;

cout << "Enter the end of the integration segment: "; // Кінець відрізку

cin >> b;

cout << "Enter the integration step: "; // Крок інтегрування

cin >> h;

start\_time = omp\_get\_wtime();

#pragma omp atomic

n = (b - a) / h; //Формула обчислення числа розбивань

#pragma omp barier

Integral = h \* (f(a) + f(b)) / 2.0;

#pragma omp for shedule(static, 4)

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

#pragma omp flush

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

cout << "The integral is approximately equal to " << Integral << "\n";

end\_time = omp\_get\_wtime();

cout << "Code execution time = " << end\_time - start\_time;

return 0;

}

case 2:

#pragma omp parallel private (Integral, n) shared(a,b,h)

{

omp\_lock\_t lock;

double start\_time, end\_time;

double Integral;

double a, b; // Задаю відрізок інтегрування a=0.0 b=1.0

double h; // Задаю крок інтегрування 0.1

double n; // Число розбивань

cout << "Enter the beginning of the integration segment: "; // Початок відрізку

cin >> a;

cout << "Enter the end of the integration segment: "; // Кінець відрізку

cin >> b;

cout << "Enter the integration step: "; // Крок інтегрування

cin >> h;

start\_time = omp\_get\_wtime();

omp\_init\_lock(&lock);

#pragma omp parallel

{

n = (b - a) / h; //Формула обчислення числа розбивань

omp\_set\_lock(&lock);

Integral = h \* (f(a) + f(b)) / 2.0;

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

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

}

cout << "The integral is approximately equal to " << Integral << "\n";

omp\_unset\_lock(&lock);

}

omp\_destroy\_lock(&lock);

end\_time = omp\_get\_wtime();

cout << "Code execution time = " << end\_time - start\_time;

return 0;

}

}

}

**Результат виконання:**

|  |
| --- |
|  |

**Висновок:** під час виконання цієї лабораторної роботи, я ознайомився з синхронізацією між паралельними потоками OpenMP програм. Зрозумів різницю між явною і неявною реалізацією та реалізував програму у двох видах синхронізації з можливістю вибору.