

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 початкової 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 case - 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 відрізка
                cin >> a;
                cout << "Enter the end of the integration segment: "; // Кінець відрізка
                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 barrier
                Integral = h * (f(a) + f(b)) / 2.0;

                #pragma omp for schedule(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();
            }
        }
    }
```

```

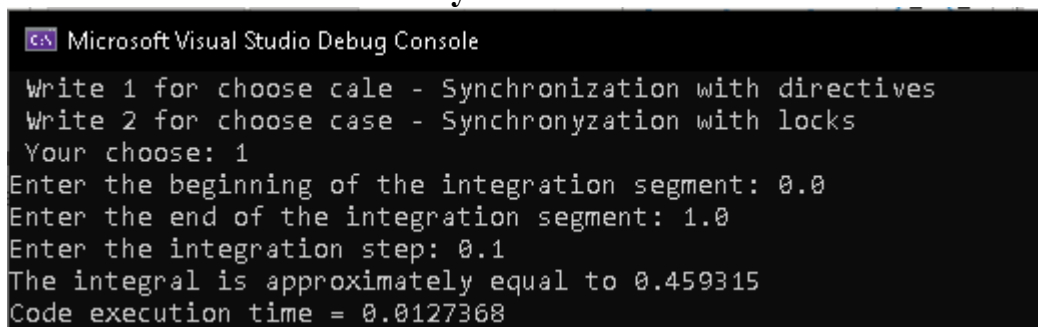
        cout << "Code execution time = " << end_time - start_time;
        return 0;
    }
    case 2:
#pragma omp parallel private (Integral, n) shared(a,b,h)
    {
        omp_lock_t lock;
        double start_time, end_time;
        double Integral;
        double a, b; // Задаю відрізок інтегрування a=0.0 b=1.0
        double h; // Задаю крок інтегрування 0.1
        double n; // Число розбивань

        cout << "Enter the beginning of the integration segment: "; // Початок відрізка
        cin >> a;
        cout << "Enter the end of the integration segment: "; // Кінець відрізка
        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:



```

Microsoft Visual Studio Debug Console
Write 1 for choose case - Synchronization with directives
Write 2 for choose case - Synchronyzation with locks
Your choose: 1
Enter the beginning of the integration segment: 0.0
Enter the end of the integration segment: 1.0
Enter the integration step: 0.1
The integral is approximately equal to 0.459315
Code execution time = 0.0127368

```

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.