МІНІСТЕРСТВО ОСВІТИ ТА НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Кафедра інформаційних систем та мереж



Лабораторна робота №2

з дисципліни

«Технології розподілених систем та паралельних обчислень»  
на тему

**«Розпаралелювання програмного коду в OpenMP»**

Варіант 25(10)

Виконав:

студент групи КН–47

Шандра О.С.

Прийняв:

к.т.н., доцент каф. ІСМ

Якушев В.С.

Львів–2019

**Мета роботи:** ознайомитись з засобами OpenMP для розпаралелювання коду та навчитись розробляти та виконувати відповідні програми.

**Теоретичні відомості**

Одним зі стандартів для програмування систем з загальною пам'яттю є інтерфейс (API) OpenMP (Open Multi-Processing). Цей стандарт реалізовано для мов програмування C, C++ та Fortran на великій кількості комп'ютерних архітектур, включаючи платформи Unix та Microsoft Windows.

OpenMP API складається з набору директив компілятора pragma (прагм), функцій та змінних середовища, що впливають на поведінку паралельної програми під час її виконання.

Для активації підтримки прагм OpenMP в компіляторі необхідно застосовувати додаткові параметри або прапорці компіляції. Прагми OpenMP починаються зі слів ***#pragma omp*** і мають наступний формат:

#pragma omp <директива> [список параметрiв]

Прототипи функцій знаходяться в файлі omp.h.

**int omp\_get\_thread\_num(void)** - функція повертає номер потоку, в якому була викликана.

**void omp\_set\_num\_threads(іnt)** - використовується для встановлення кількості потоків, що будуть виконуватись в наступному паралельному блоці.

**іnt omp\_get\_num\_threads(void)** - визначення поточної кількості паралельних потоків.

**іnt omp\_in\_paгallel(void)** - визначує чи виконується блок коду в паралельному режимі.

Як правило, OpenMP застосовується для розпаралелювання циклів. Однак, в OpenMP є також засоби для підтримки паралелізму на рівні функцій. Цей механізм називається секціями OpenMP (OpenMP sections). Для створення паралельного блоку секцій застосовується директива #pragma omp sections, або її скорочений варіант #pragma omp parallel sections. Секції в блоці створюються директивою #pragma omp section. Кожній секції ставиться у відповідність один потік, і всі секції виконуються паралельно.

**Завдання**

1. Виконати програми, що наведені в лістингах та проекспериментуйте з ними. Зокрема в програмі реалізуйте процедуру введення кількості потоків користувачем. Реалізуйте процедуру в якій парні потоки друкують «Hello, OpenMP!», а непарні – ім'я та прізвище студента.

2. Розробити програму згідно з варіантом та передбачити в ній використання двох потоків кожен з яких виконує інші операції. Для створення потоків використовувати механізм секцій.

**Хід роботи**

#include <stdio.h>

#include <iostream>

#include <omp.h>

using namespace std;

int main(int argc, char \*argv[]){

setlocale( LC\_ALL,"UKR");

int num\_threads;

cout << "Введiть кiлькiсть потокiв: ";

cin >> num\_threads;

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

#pragma omp parallel

{

if (omp\_get\_thread\_num() % 2 == 0)

printf("\nПотiк №%3d: Hello, OpenMP!", omp\_get\_thread\_num());

else

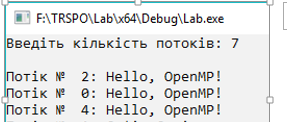
printf("\nПотiк №%3d: Шандра Олег", omp\_get\_thread\_num());

}

getch();

return 0;

}



*Рис.1.Результат виконання завдання №1*

using System;

using System.Linq;

using System.Threading.Tasks;

namespace Lab2

{

class Program

{

static void Main(string[] args)

{

//Варiант 25

int[] Ai = new int[] { 83, -2, 5654, 76, 67, 54, 76, 2333, 4, 5, -13, 54, -43, 653, -123, 5, 854, 12, 54, 13, 64, 432, 54, 234, 543, -3444, 763, 153, 124, 653 };

var thread1 = new Task<int>(() =>

{

Console.WriteLine("Обрахунок додатнiх...");

var res = Ai.Count(i => i > 0);

Console.WriteLine("К-сть додатнiх k1 = " + res);

return res;

});

var thread2 = new Task<int>(() =>

{

Console.WriteLine("Обрахунок вiд'ємних...");

var res = Ai.Count(i => i < 0);

Console.WriteLine("К-сть вiд'ємних k2 = " + res);

return res;

});

thread1.Start();

thread2.Start();

var k1 = thread1.Result;

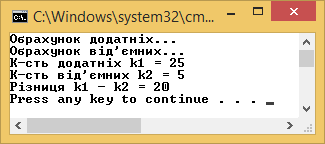
var k2 = thread2.Result;

Console.WriteLine("Рiзниця k1 - k2 = " + (k1 - k2));

}

}

}



*Рис.2.Результат виконання завдання №2*

**Висновок**

На даній лабораторній роботі було вивчено засобами OpenMP для розпаралелювання коду та як розробляти та виконувати відповідні програми. В ході виконання було застосовано механізм секцій.