**Лабораторна робота №5**

Шевченко Н. В.

**Тема:** Технологія OpenMP.

**Мета:** Познайомитись з базовими директивами OpenMP та навчитись їх застосовувати для розпаралелювання послідовних програм.

Програма починається з підключення необхідних бібліотек: iostream для операцій введення/виведення та omp.h для використання OpenMP (Open Multi-Processing) - бібліотеки для паралельних обчислень. Далі визначаються дві константи rows та cols, які задають розміри двовимірного масиву arr розміром 100000 на 1000.

Функція init\_arr() призначена для ініціалізації масиву arr випадковими числами від 0 до 99. Спочатку вона ініціалізує генератор випадкових чисел за допомогою srand(time(0)), де time(0) повертає поточний час у секундах, який використовується як початкове значення для генератора. Потім у двох вкладених циклах проходиться по всіх елементах масиву та присвоюється випадкове число від 0 до 99 за допомогою виразу rand() % 100.

Функція get\_sum(int num\_threads, double& execution\_time) обчислює суму всіх елементів масиву arr з використанням вказаної кількості потоків num\_threads. Спочатку визначається змінна sum для зберігання суми та t1 для запам'ятовування поточного часу за допомогою omp\_get\_wtime(). Далі використовується директива #pragma omp parallel for з параметром num\_threads(num\_threads), яка розподіляє ітерації зовнішнього циклу між вказаною кількістю потоків. Для кожної ітерації зовнішнього циклу виконується внутрішній цикл, в якому обчислюється сума елементів відповідного рядка. Ключове слово reduction(+:sum) вказує, що змінна sum повинна бути проредукована (складена) між потоками після завершення паралельного регіону. Після завершення паралельного регіону, обчислюється час виконання execution\_time як різниця між поточним часом omp\_get\_wtime() та початковим t1. Нарешті, функція повертає обчислену суму sum.

Функція get\_min\_row(int num\_threads, int& min\_row, int& min\_sum, double& execution\_time) знаходить рядок масиву arr з найменшою сумою елементів, використовуючи вказану кількість потоків num\_threads. Спочатку ініціалізується змінна min\_sum максимальним значенням INT\_MAX, а min\_row невизначеним значенням -1. Потім запам'ятовується поточний час у змінній t1. Далі використовується директива #pragma omp parallel for num\_threads(num\_threads), яка розподіляє ітерації зовнішнього циклу між вказаною кількістю потоків. Для кожної ітерації зовнішнього циклу обчислюється сума елементів відповідного рядка у змінній row\_sum. Якщо row\_sum менше за поточну min\_sum, то потрібно оновити min\_sum та min\_row. Однак, оскільки кілька потоків можуть одночасно намагатися оновити ці змінні, виникає ризик конфлікту потоків. Тому оновлення відбувається у критичній секції #pragma omp critical, яка гарантує, що лише один потік одночасно може виконувати код всередині цієї секції. Після завершення паралельного регіону, обчислюється час виконання execution\_time та функція повертає оновлені значення min\_row та min\_sum.

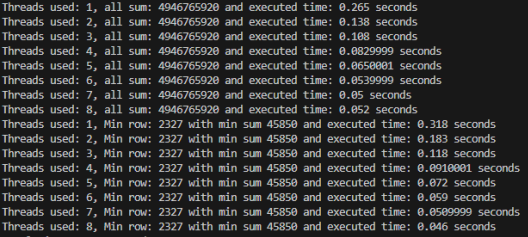
У функції main() спочатку визначається константа max\_thread зі значенням 8, яка задає максимальну кількість потоків для використання. Далі оголошуються масиви для зберігання результатів обчислень для різної кількості потоків: sum для сум, min\_sum для найменших сум рядків, min\_row для індексів рядків з найменшими сумами та executed\_time для часу виконання.

Викликається функція init\_arr() для ініціалізації масиву arr. Потім встановлюється налаштування omp\_set\_nested(1), яке дозволяє використовувати вкладені паралельні регіони.

Запам'ятовується поточний час у змінній t1 перед початком паралельних обчислень. Далі використовується директива #pragma omp parallel sections, яка створює дві незалежні паралельні секції. У першій секції знаходиться цикл, який для кожної кількості потоків від 1 до max\_thread викликає функцію get\_sum() та зберігає результат у відповідному елементі масиву sum, а також час виконання у відповідному елементі масиву executed\_time[0]. У другій секції знаходиться цикл, який для кожної кількості потоків від 1 до max\_thread викликає функцію get\_min\_row() та зберігає результати у відповідних елементах масивів min\_row, min\_sum та executed\_time[1].

Після завершення паралельних обчислень, запам'ятовується поточний час у змінній t2. Далі у двох циклах виводяться результати обчислень для кожної кількості потоків: сума всіх елементів масиву, найменша сума рядка, індекс рядка з найменшою сумою та час виконання для кожного обчислення.

Нарешті, виводиться загальний час виконання програми як різниця між t2 та t1.



**Висновок**

У цій лабораторній роботі були вивчені основи технології OpenMP для розпаралелювання програм. Розглянутий приклад продемонстрував прискорення обчислень за рахунок розподілу навантаження між потоками. Однак існує межа, після якої додавання нових потоків не дає значного приросту продуктивності. Також навчилися уникати конфліктів потоків при спільному доступі до даних за допомогою критичних секцій. OpenMP є зручним інструментом для паралельних обчислень, який потребує обережного використання для запобігання потенційних проблем.