Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3086 Programación de microprocesadores

Ciclo 2 de 2,024
Temario A

Laboratorio 05

Osman Emanuel de León García - 23428

Competencias para desarrollar

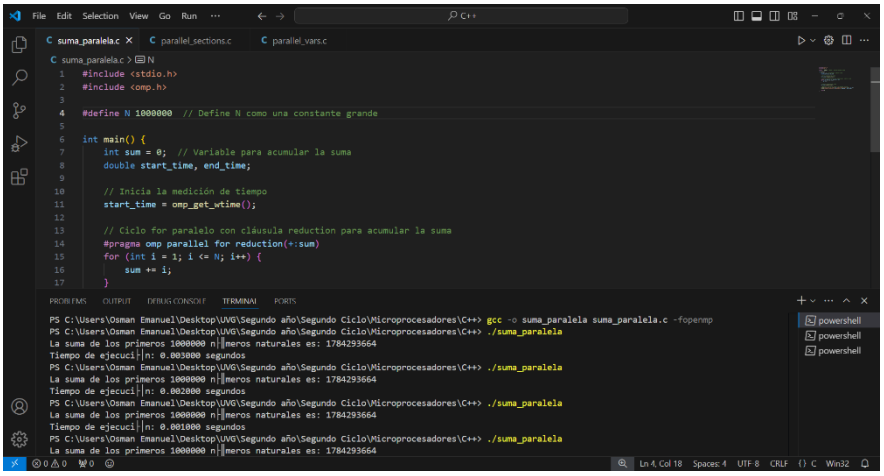
Distribuir la carga de trabajo entre hilos utilizando programación en C y OpenMP.

Instrucciones

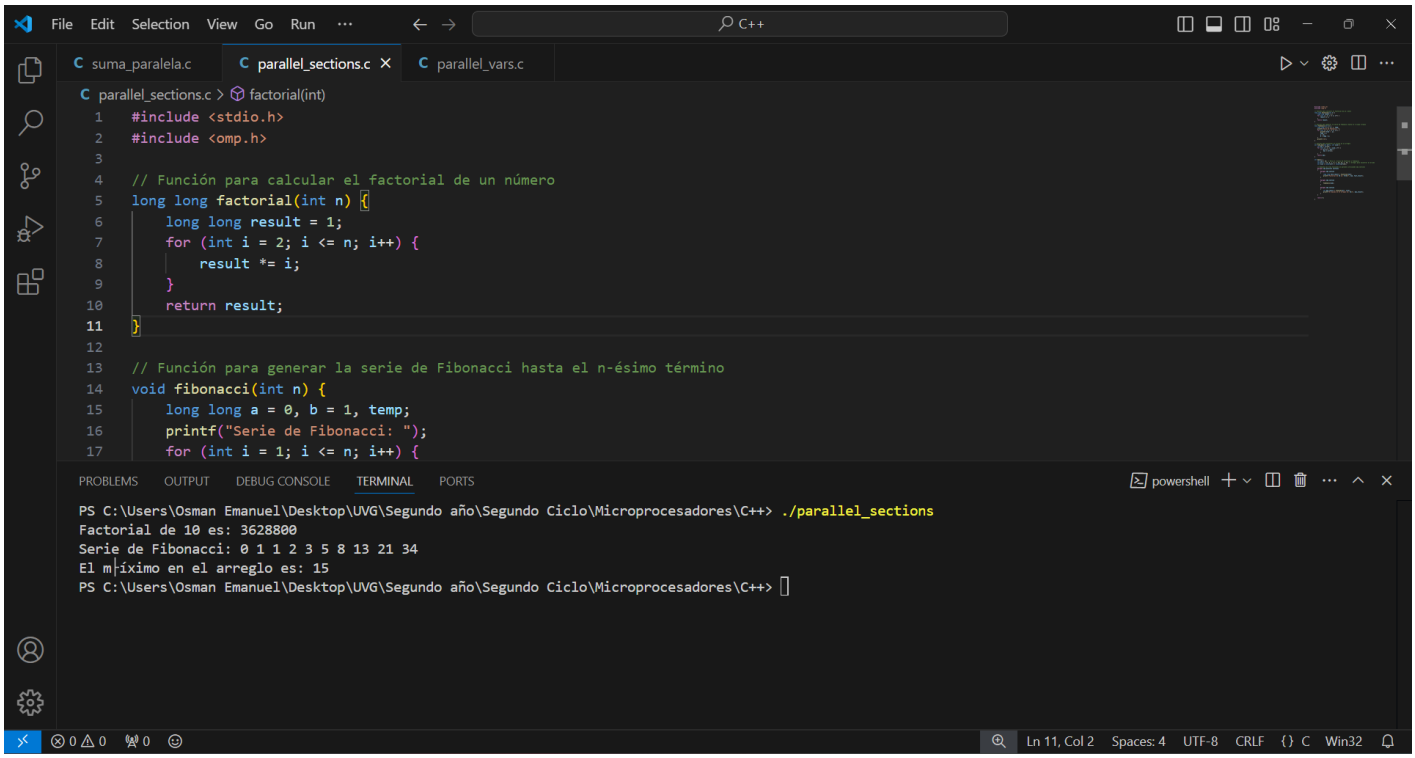
Esta actividad se realizará individualmente. Al finalizar los períodos de laboratorio o clase, deberá entregar este archivo en formato PDF y los archivos .c en la actividad correspondiente en Canvas.

1. **(18 pts.)** Explica con tus propias palabras los siguientes términos:
   a) Private: significa que cada hilo tiene su propia copia de esa variable. Los cambios hechos por un hilo no afectan las copias de los otros hilos.
   b) Shared: Es una variable compartida entre todos los hilos en OpenMP. Todos los hilos acceden y pueden modificar la misma instancia de la variable.
   c) Firstprivate: Similar al private, pero las variables comienzan con el mismo valor que tenían antes de entrar en la región paralela. Cada hilo tiene su propia copia inicializada con ese valor.
   d) Barrier: Es una sincronización en OpenMP donde todos los hilos deben detenerse y esperar hasta que todos lleguen a este punto antes de continuar. Asegura que todos los hilos alcancen la barrera antes de avanzar.
   e) Critical: Es una sección en OpenMP donde solo un hilo puede ejecutar el código a la vez. Se usa para proteger secciones de código que acceden a recursos compartidos para evitar condiciones de carrera.
   f) Atomic: Como critical pero mas ligero Es una sección en OpenMP donde solo un hilo puede ejecutar el código a la vez. Se usa para proteger secciones de código que acceden a recursos compartidos para evitar condiciones de carrera.

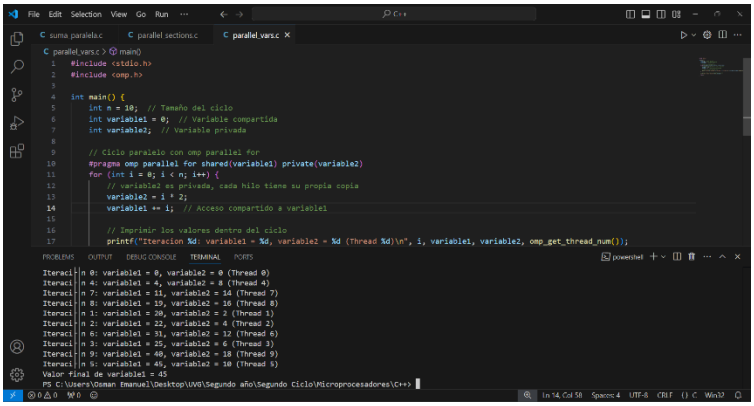
2. **(12 pts.)** Escribe un programa en C que calcule la suma de los primeros N números naturales utilizando un ciclo ***for*** **paralelo**. Utiliza la cláusula **reduction con +** para acumular la suma en una variable compartida.
   a) Define N como una constante grande, por ejemplo, N = 1000000.
   b) Usa omp_get_wtime() para medir los tiempos de ejecución.



3. **(15 pts.)** Escribe un programa en C que ejecute tres funciones diferentes en paralelo usando la **directiva #pragma omp sections**. Cada sección debe ejecutar una función distinta, por ejemplo, una que calcule el factorial de un número, otra que genere la serie de Fibonacci, y otra que encuentre el máximo en un arreglo, operaciones matemáticas no simples. Asegúrate de que cada función sea independiente y no tenga dependencias con las otras.



4. **(15 pts.)** Escribe un programa en C que tenga un ciclo for donde se modifiquen dos variables de manera paralela usando #pragma omp parallel for.
   a. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
   b. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
   c. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.



Diferencias entre clausulas

1. **Cláusula shared**:
   - **Comportamiento**: variable1 es compartida entre los hilos. Dado que todos los hilos pueden acceder a ella simultáneamente, puede haber interferencias (condiciones de carrera), lo que resulta en un valor final inesperado o incorrecto de variable1 después del ciclo. La suma acumulada podría no ser precisa debido a la falta de sincronización.
2. **Cláusula private**:
   - **Comportamiento**: variable2 es privada para cada hilo, lo que significa que cada hilo tiene su propia copia de variable2 y las modificaciones realizadas por un hilo no afectan a las copias de otros hilos. Esto garantiza que no haya interferencias entre los hilos y cada hilo puede modificar variable2 independientemente.

Por tanto

- **variable1**: Debido a que es compartida, el valor final puede no ser el que se esperaría de un ciclo secuencial, debido a las condiciones de carrera.
- **variable2**: Cada hilo tendrá su propia versión, y no habrá interferencias entre hilos.

5. **(30 pts.)** Analiza el código Ejercicio_05C, encuentra todas las posiciones en el vector DBin en las cuales aparece un conjunto de claves (contenidas en el vector keys). Las posiciones donde las claves aparecen se almacenan en un nuevo vector DBout (el orden en DBout de las posiciones encontradas es irrelevante).
   Escribe un programa con OpenMP que implemente una estrategia de paralelización, siguiendo un modelo de ejecución productor-consumidor. El código del productor y del consumidor debe estar en dos tareas diferentes.

6. **(10 pts.) REFLEXIÓN DE LABORATORIO: se habilitará en una actividad independiente.**

Link Repositorio Github: https://github.com/Emadlgg/Lab5.git