# Laboratorio 05

***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:
2. Private
   1. Las variables declaradas como private son exclusivas para cada hilo. Cada hilo tiene su propia copia de la variable, y los cambios realizados por un hilo no afectan a las copias de otros hilos.
3. Shared
   1. Las variables declaradas como shared son compartidas entre todos los hilos. Todos los hilos pueden acceder y modificar la misma variable, lo que puede llevar a condiciones de carrera si no se maneja adecuadamente.
4. Firstprivate
   1. Las variables declaradas como firstprivate son similares a las private, pero además se inicializan con el valor de la variable original antes de la región paralela. Cada hilo obtiene una copia privada inicializada con el valor de la variable original.
5. Barrier
   1. Una barrier es un punto de sincronización donde todos los hilos deben esperar hasta que todos hayan llegado a ese punto antes de continuar. Esto asegura que todos los hilos han completado una parte específica del código antes de proceder.
6. Critical
   1. Una sección critical asegura que solo un hilo puede ejecutar el bloque de código asociado a la vez. Esto es útil para proteger secciones de código que acceden a recursos compartidos y evitar condiciones de carrera.
7. Atomic
   1. Atomic asegura que una operación específica en una variable se realiza de manera atómica, es decir, sin interrupciones. Esto es útil para operaciones simples como incrementos o actualizaciones que deben ser realizadas de manera segura en un entorno paralelo.
8. **(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.
9. Define N como una constante grande, por ejemplo, N = 1000000.
10. Usa omp\_get\_wtime() para medir los tiempos de ejecución.
11. **(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.
12. **(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.
    1. Usa la cláusula shared para gestionar el acceso a la variable1 dentro del ciclo.
    2. Usa la cláusula private para gestionar el acceso a la variable2 dentro del ciclo.
    3. Prueba con ambas cláusulas y explica las diferencias observadas en los resultados.
13. **(30 pts.)** Analiza el código en el programa Ejercicio\_5A.c, que contiene un programa secuencial. Indica cuántas vseces aparece un valor key en el vector a. Escribe una versión paralela en OpenMP utilizando una descomposición de tareas **recursiva**, en la cual se generen tantas tareas como hilos.
14. **REFLEXIÓN DE LABORATORIO: se habilitará en una actividad independiente.**