

## 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:
  - a) `private`
    - a. Hace que cada subproceso tenga su propia instancia de un variable.
  - b) `shared`
    - a. Recalca que solo una o muchas variables sean compartidas entre los diferentes subprocesos.
  - c) `firstprivate`
    - a. Fija que cada subproceso tenga su propia instancia de una variable y así cada variable debe de ser inicializada con el valor de la variable, ya que esta fue creada antes de la construccion paralela.
  - d) `barrier`
    - a. Es una sincronización que asegura que todos los hilos de un equipo alcancen un punto específico en el código antes de que cualquiera de ellos pueda continuar.
  - e) `critical`
    - a. Restringe la ejecucion del bloque estructurado asociado a un unico hilo a la vez.
  - f) `atomic`
    - a. Es para especificar la ubicacion de memoria que será actualizada atómicamente.
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.

5. **(30 pts.)** Analiza el código en el programa Ejercicio\_5A.c, que contiene un programa secuencial. Indica cuántas veces 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.
  
6. **REFLEXIÓN DE LABORATORIO:** se habilitará en una actividad independiente.