Santiago Chitiva Contreras

**Implementación del Algoritmo Clásico de Multiplicación de Matrices**

Primero que todo, se reservó un bloque de memoria estática utilizando la constante "RESERVA", que define el tamaño de dicho espacio bloque. Este espacio se utilizó para almacenar las dos matrices de entrada (A y B) y la matriz de salida o resultado (C).

Luego de esto, se crearon punteros para las matrices de tipo double, que apuntan a las ubicaciones de memoria ya reservadas y se asigna la memoria a las matrices utilizando esos punteros.

Después, se solicita al usuario que ingrese la dimensión de las matrices y el número de hilos que va a usar para la multiplicación de matrices. Se validan los argumentos de entrada proporcionados por el usuario, verificando que sean números positivos (mayores a 0), y si no se proporcionan los argumentos esperados, se muestra un mensaje de error y se termina la ejecución del programa.

Seguido a esto se inicializaron las matrices de entrada con valores predefinidos y se imprimió su contenido para verificar que hayan sido inicializadas correctamente. Sin embargo, para organizar y reutilizar mejor el código, se crearon dos funciones por aparte, una para la inicialización (initMatrices), donde cada elemento de las matrices se calcula en base a su índice y se almacena en el bloque de memoria reservado; y otra para la impresión (imprMatrices), en la cual, si la dimensión de las matrices es menor que 9, se imprimen todos los elementos; de lo contrario, se muestra un mensaje indicando que las matrices son demasiado grandes para ser impresas.

Cuando ya tenemos todo esto, se implementa el Algoritmo Clásico de multiplicación de matrices utilizando bucles anidados para calcular cada elemento de la matriz resultante y se verifica el resultado de la multiplicación de matrices. No obstante, al igual que con los casos anteriores, también se creó una función por aparte (multiMatrices) que recibe como argumentos las matrices de entrada (mA y mB), así como la matriz resultado (mC). Específicamente, esta función usa tres bucles anidados para realizar la multiplicación de matrices, calculando cada elemento de la matriz resultante y almacenándolo en mC. Finalmente se imprime la matriz resultante (mC) después de la multiplicación y se muestra un mensaje indicando el fin del programa.

Ahora bien, adentrándonos en el tema de la clase, se declaró un arreglo de hilos que se utiliza para ejecutar la multiplicación de matrices de forma concurrente y se crearon hilos utilizando la biblioteca PTHREAD. Se utilizó un bucle "for" para crear múltiples hilos, cada uno de los cuales ejecuta la función "multiMatrices" de manera concurrente para multiplicar las matrices.

Por último, se definió una estructura de datos para encapsular los argumentos de entrada necesarios para la función de multiplicación de matrices, lo que facilita el manejo y el paso de datos a través de los hilos.