Con respecto a la descripción del algoritmo, se tiene que el algoritmo deberá de recibir una matriz cuadrada de orden n y, el resultado será esa misma matriz ordenada siguiendo una lógica de víbora (snake-order). Más adelante explicaré a que se refiere esta lógica.

Los pasos que sigue el algoritmo para realizar el ordenamiento son bastante sencillos y el número de veces que se repiten es logaritmo base 2 de n, donde n es el orden de la matriz. Los pasos a seguir se dividen en dos fases principales: La primera es la fase de ordenamiento de filas y la segunda la fase de ordenamiento de columnas.

En la fase de ordenamiento de filas, si se tiene una fila con número par, entonces esta se ordenará de manera ascendente de izquierda a derecha. En el caso de que se tenga una fila con número impar, entonces esta se ordenará de manera ascendente de derecha a izquierda. Ya que se realizó lo anterior para todas las filas de la matriz, se continua a la fase de ordenamiento de columnas. En esta fase, todas las columnas se ordenarán de manera ascendente, de arriba hacia abajo. Ya que se repitió este proceso, se tendrá una matriz ordenada siguiendo una lógica de serpiente. Para entender mejor esta lógica y el proceso, observemos un pequeño ejemplo. ***\*CAMBIO DE DIAPOSITIVA\****

El código secuencial se realizó en C y el código paralelo se realizó en OpenMP, para poder comparar de mejor manera los códigos. ***\*CAMBIO DE DIAPOSITIVA\****

En cuanto al código secuencial, se empieza por importar algunas bibliotecas que contienen funciones que usaremos y definimos las funciones a utilizar. En el método main, se le pide al usuario que ingrese el orden de la matriz cuadrada y esta se rellenará con números aleatorios con valores entre 0 y 99.

Respecto a las funciones que realizarán el proceso de ordenamiento, se encuentran next() y sort(). En la función next() se encuentran las fases de ordenamiento de columnas y de filas, siguiendo los pasos descritos con anterioridad. En la función sort(), se encuentra un algoritmo de ordenamiento interno, que en este caso es Bubble Sort. Este algoritmo de ordenamiento fue escogido para la versión secuencial ya que, cuando se realiza la versión paralela, se puede paralelizar el Bubble Sort, teniendo así un algoritmo de ordenamiento llamado como Odd-Even Sort, el cual se detallará más adelante.

Por último, se tienen las funciones que nos ayudan a llenar e imprimir el arreglo. Dado que se está trabajando con memoria dinámica en C, es necesario liberarla, y por ello está la función liberarMemoria.

Ahora, con respecto a la versión paralela, existían dos formas de realizar la paralelización de este código: Una se encargaba de realizar la paralelización del método de ordenamiento Bubble Sort y la otra se encargaba de realizar la paralelización en el proceso que se sigue para ordenar la matriz.

Si se intentaban realizar las dos paralelizaciones al mismo tiempo, se encontraba que era menos eficiente a que si solo empleaba alguna de ellas, puesto que la sincronización y el balance de trabajo llegaban a ser bastante complicados de realizar. De igual manera, se encontró que la mejor paralelización que se podía realizar era la del proceso de ordenamiento de la matriz. ***\*CAMBIO DE DIAPOSITIVA\****

En la paralelización del ordenamiento de la matriz, la región a paralelizar era aquella que se encontraba en la función next(). Para realizar esta paralelización, se utilizaron las directivas for y barrier. Con la directiva for se dividía la matriz cuadrada para que un conjunto de hilos trabajara sobre cierto número de columnas y filas, pues estas son independientes entre sí. La directiva barrier se usaba para esperar a que todos los hilos terminaran una respectiva fase, pues si no se realizaba esto, no se llegaba al correcto ordenamiento de la matriz.

***\*CAMBIO DE DIAPOSITIVA\****

Por otro lado, en la paralelización del Bubble Sort, se tiene el algoritmo de ordenamiento Odd-Even Sort. Este algoritmo de ordenamiento consiste en dividir el ordenamiento en dos fases: Una fase impar y otra par. En ambas fases, todos los números se irán ordenando en parejas. En la fase impar, se empezarán a contar las parejas desde el primer elemento en la lista. En la fase par, se empezarán a contar las parejas desde el segundo elemento en la lista. Este proceso se repite n veces, donde n es el tamaño de la lista.

En cuanto al código, la región paralela se encontraba dentro de la función sort, como se muestra a continuación. Igual se hizo uso de la directiva for, para que cada hilo trabajase con una cierta cantidad de parejas de números en la lista.

***\*CAMBIO DE DIAPOSITIVA\****