Tarea 2

Diego Huerta diego.huertan@alumnos.uv.cl

Vicente Miralles Vicente.miralles@alumnos.uv.cl

Matias Rivas matias.rivas@alumnos.uv.cl

1. Introducción

En el ámbito de la programación concurrente, la implementación eficiente de algoritmos es esencial para aprovechar al máximo los recursos de un sistema. En este contexto, se creará un programa que surgirá como una solución multiproceso diseñada para generar un histograma de palabras a partir de un archivo de texto. Esta aplicación, desarrollada en C++, se fundamenta en la solución secuencial previamente proporcionada. La premisa principal de este proyecto es asegurar que la versión multi-thread produzca resultados idénticos a la solución secuencial

2. Materiales y Métodos

Esta sección muestra ejemplos de uso de secciones, subsecciones, figuras, tablas y ecuaciones.

2.1. Materiales

Se utilizaron datos provenientes de texto que contiene una parte de él libro don quijote

2.2. Métodos

Se utilizaron múltiples métodos, como el multithreading y el procesamiento de texto con mutex, para abordar de manera efectiva la tarea. El multithreading permitió dividir la tarea en múltiples subprocesos que trabajaban simultáneamente, lo que aceleró el procesamiento de datos y mejoró la eficiencia. Por otro lado, el uso de mutex garantizó que múltiples hilos no interfirieran entre sí al acceder y modificar los mismos datos, lo que resultó en un procesamiento de texto más seguro y coherente. La combinación de estos enfoques permitió una mejor utilización de los recursos del sistema y una mayor velocidad en la ejecución de la tarea.

3. Resultados

3.1. Diagrama de flujo

Él siguiente diagrama de flujo describe él funcionamiento de la software implementado, considerando desde que se ingresan los parámetros por consola. Cabe destacar que él diagrama de flujo es secuencial por ende no muestra él funcionamiento multi-proceso del programa.

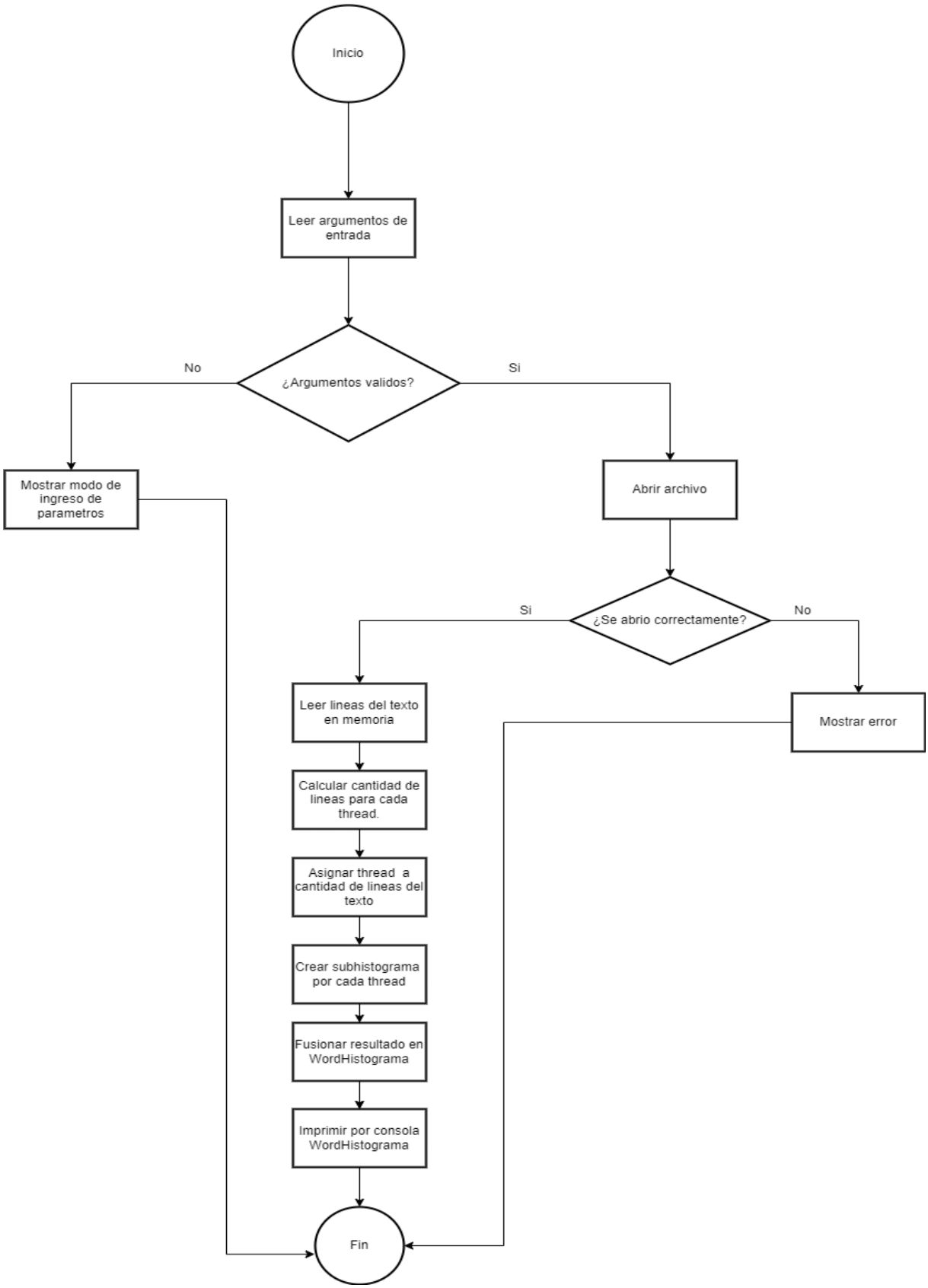
Inicio
Leer argumentos de entrada
¿Argumentos validos?
No
Si
Mostrar modo de ingreso de parametros
Abrir archivo
¿Se abrio correctamente?
Si
No
Leer lineas del texto en memoria
Mostrar error
Calcular cantidad de lineas para cada thread.
Asignar thread a cantidad de lineas del texto
Crear subhistograma por cada thread
Fusionar resultado en WordHistograma
Imprimir por consola WordHistograma
Fin

Figura 1. Diagrama de flujo

3.2. Diseño Multi-Procesos

La estrategia utilizada para implementar múltiples threads en él programa se consideró él uso de mutex para evitar problemas de sincronización entre los threads además se compartió una función general entre los threads la cual es `procesarlineas.`

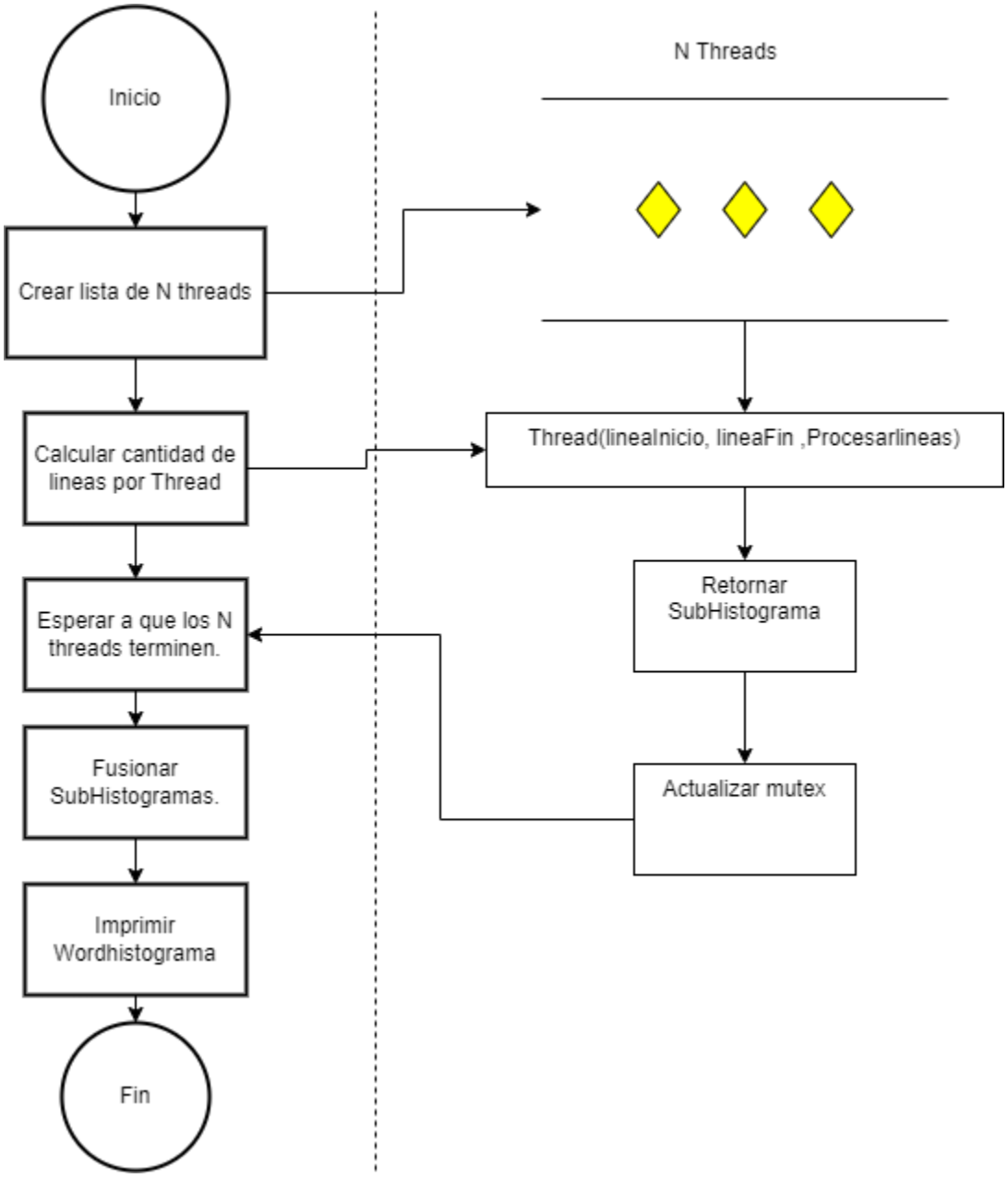

Figura 2. Diagrama Multi-thread

4. Discusión y conclusiones

Este proyecto demuestra la eficacia de la programación concurrente mediante el uso de multithreading y mutex. La implementación multiproceso aceleró significativamente el procesamiento de un extenso archivo de texto al dividir la carga de trabajo entre varios hilos, garantizando resultados coherentes.

Este enfoque tiene amplias aplicaciones en problemas que requieren procesamiento paralelo de datos, especialmente cuando se deben evitar conflictos al acceder y modificar datos compartidos por múltiples hilos.