Tecnológico de Monterrey

Instituto Tecnológico y de Estudios Superiores de Monterrey
Campus Puebla

Actividad Integradora 5.3 Resaltador de sintaxis paralelo

Implementación de métodos computacionales

(Gpo 570)

Alumno:

Jesús Jiménez Aguilar

A01735227

Profesor:

PhD. Pedro Oscar Pérez Murueta

26 de julio de 2022

a) Reflexión sobre la solución planteada, los algoritmos implementados y sobre el tiempo de ejecución de estos.

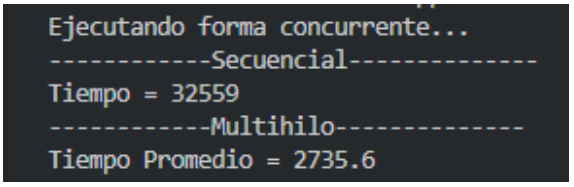
En la solución planteada se implementó un algoritmo que se encarga de la lectura y clasificación en C++ de un código de C++ para después re-escribirlo y categorizarlo de manera visual a través de una gama de colores especificados en un documento CSS conectado a una página HTML que procede a conectar con la información. Para esto se utilizó como base el resaltador léxico realizado previamente para la evidencia anterior.

El trabajo producido para este código es eficiente y minucioso, pero la complejidad del algoritmo implementado varía según la función a ejecutar, ya que en las funciones encargadas de determinar que string hace match con la función declarada (isXXXXX) tienen como complejidad temporal O(1) o constante para todos los casos, ya que presenta un óptimo tiempo de ejecución menor a un segundo en cualquiera de los casos ejecutados. En el caso de la función para crear arreglos de cada palabra separada para que de esta forma el resaltador pueda trabajar de manera sencilla recorriendo n*n en el peor de los casos, tiene como complejidad temporal O(n^2). Tanto el resaltador secuencial como el concurrente tienen como complejidad temporal O(n^4), ya que recorren n pasos 4 veces para el peor de los casos.

En referencia a los tiempos de prueba, se ejecutaron 3 casos de prueba donde se obtuvieron los siguientes resultados:

- **Caso de prueba 1:**

Utilizando 9 archivos y 8 hilos tenemos los siguientes resultados:



Secuencial = 32559 ms Concurrente = 2735.6 ms

Speed up: $S_8 = \frac{T_1}{T_8} | S_8 = \frac{32559}{2735.6} | S_8 = 11.901$

Lo anterior implica que utilizando la programación concurrente con 8 hilos para 9 archivos es casi 12 veces más rápida que la secuencial.

- **Caso de prueba 2:**

Utilizando solo un archivo con 8 hilos de trabajo, se tienen los siguientes resultados

```
Ejecutando forma secuencial...
Ejecutando forma concurrente...
------------Secuencial--------------
Tiempo = 1911
------------Multihilo--------------
Tiempo Promedio = 1952
```

Secuencial = 1911 ms Concurrente = 1952 ms

Speed up: $S_8 = \frac{T_1}{T_8} | S_8 = \frac{1911}{1952} | S_8 = 0.02561 \ | \ S_1 = \frac{T_8}{T_1} | S_1 = \frac{1952}{1911} | S_1 = 1.021$

Esto significa no hay un speed up cuando se utiliza un solo archivo, de hecho es más rápido utilizar la manera secuencial teniendo un speed up de 1.021

- **Caso de prueba 3:**

Utilizando 22 archivos y 8 hilos obtenemos los siguientes resultados:

```
Ejecutando forma secuencial...
Ejecutando forma concurrente...
------------Secuencial--------------
Tiempo = 90562
------------Multihilo--------------
Tiempo Promedio = 4007.9
```

Secuencial = 90562 ms Concurrente = 4007.9 ms

Speed up: $S_8 = \frac{T_1}{T_8} | S_8 = \frac{90562}{4007.9} | S_8 = 22.59$

Esto quiere decir que para 22 archivos utilizando 8 hilos utilizando la programación concurrente obtenemos que es 22 veces más rápido que la secuencial

La realización de este proyecto me ha ayudado a poder contrastar de excelente manera la eficiencia del tiempo que la implementación de recursos como el paralelismo puede tener. Así como gracias a los conceptos vistos en clase como los AFD, la programación paralela y la concurrente, me ayudaron a generar un resaltador léxico eficiente.

En el aspecto de las implicaciones éticas que puede tener en la sociedad este tipo de tecnologías, es que gracias a este tipo de proyectos sea posible que las nuevas generaciones de desarrolladores tengan algo en que apoyarse al momento de desarrollar un completo pensamiento crítico en referencia al desarrollo de la lógica computacional, así como lograr un aprendizaje en la diferenciación de las distintas expresiones

existentes en los lenguajes de programación para aquellos principiantes por medio de las resaltaciones de cada tipo de expresión o léxico en distintos colores pues se vuelve más amigable la programación con cualquier tipo de usuario.

También en la accesibilidad se puede observar otra implicación ética, pues es probable que una persona con diferentes capacidades visuales, como el daltonismo, presente problemas para diferenciar los colores que se asignaron cada tipo de expresión. Esto debido a que la afección provoca que algunos de los colores utilizados solamente sean vistos como sombras grises o una tonalidad totalmente opuesta a la real, esto ocurre principalmente con colores como azules, verdes y rojos.