Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Querétaro

Implementación de métodos computacionales TC2037.601

Actividad integradora 3.4:

Resaltador de sintaxis C#

Presenta:

José Ricardo Rosales Castañeda | A01709449

Dante David Perez Perez | A01709226

Querétaro, Querétaro 13 de abril, 2023

Categorías léxicas y expresiones regulares que las identifican

Palabras clave: son palabras reservadas del lenguaje, se usó una lista de con las palabras reservadas o keywords de c#

("abstract" "as" "base" "bool" "break" "byte" "case" "catch" "char" "checked"
"class" "const" "continue" "decimal" "default" "delegate" "do" "double" "else"
"enum" "event" "explicit" "extern" "false" "finally" "fixed" "float" "for"
"foreach" "goto" "if" "implicit" "in" "int" "interface" "internal" "is" "lock"
"long" "namespace" "new" "null" "object" "operator" "out" "override" "params"
"private" "protected" "public" "readonly" "ref" "return" "sbyte" "sealed" "short"
"sizeof" "stackalloc" "static" "string" "struct" "switch" "this" "throw" "true"
"try" "typeof" "uint" "ulong" "unchecked" "unsafe" "ushort" "using" "virtual"
"void" "volatile" "while")

Identificadores: son nombres que se utilizan para identificar variables, métodos, clases y otros elementos del programa, [a-zA-Z_][a-zA-Z0-9_]*$

Literales: son valores de las variables o constantes, strings y nums

Strings: (\"(\\\\.|[^\\\\\"])*\")

Nums: (\\b[0-9]+(\\.[0-9]*)?\\b)

Operadores: son símbolos que se utilizan para realizar operaciones en expresiones, como sumas, restas, multiplicaciones, comparaciones y asignaciones.

("+" "-" "*" "/" "%" "&" "|" "^" "!" "~" "=" "<" ">" "+=" "-=" "*=" "/=" "%="

"&=" "|=" "^=" "<<" ">>" ">>>" "<<=" ">>=" ">>>=" "==" "!=" "<=" ">="
"&&" "||"
"++" "--" "?" "??")

Separadores: son símbolos que se utilizan para separar elementos en el código fuente, como paréntesis, corchetes, llaves, comas y puntos y comas.
(";" "," "." "(" ")" "[" "]" "{" "}" "<" ">" ":" "::" "..." "=>" "??")

Comentarios: son textos que no se compilan, usualmente para explicar el código, multilinea y monolinea
(//.*|/*.**/|/*.*|.**/)

System: Por gusto personal se agregaron las palabras “System”, “Console” y “Program” a una lista separada.
("System" "Console" "Program")

Documentación

Objetivo

Crear una aplicación que reciba un archivo C# y devuelva un archivo HTML que resalte toda la sintaxis del lenguaje utilizando Racket

Algoritmo

1. Se define una serie de listas de palabras clave (keywords), operadores, separadores, nombres de clase y otros elementos que se encuentran en el lenguaje C#.
2. Luego, se definen una serie de expresiones regulares para identificar tokens como números, cadenas de caracteres, comentarios y nombres de variables.
3. La función "categorize-token" tiene como propósito tomar un token como input y generar una cadena que incluya el token, enmarcado por etiquetas HTML que señalan la categoría a la que pertenece el token (por ejemplo, "keyword", "operador", "número", "cadena", entre otros).
4. Se define otra función llamada “replace-all-tokens” recorre la cadena de entrada carácter por carácter y divide la cadena en tokens separados por espacios, operadores y otros caracteres especiales.
5. Se define la función “categorize-token” que hace que cada token se clasifique y se agrega a una lista de tokens. Al final de cada línea, se agrega la lista de tokens a otra lista de líneas, que se utilizará más adelante para construir la salida HTML.

Reflexión

Dante Pérez: Sobre el algoritmo que implementamos se reduce en separar los strings en una lista de caracteres para poder después darle una categoría específica usamos, podríamos poder utilizado *string->list, ya que* esta separa el string en una lista de caracteres, otra cosa que pudimos haber mejorado es hacer que el código sea *O(n)* por el hecho de que haría que el tiempo demorara menos que el actual esto por las iteraciones que realizamos, por otro lado, el tiempo de ejecución que se tardó fue de 5s esto porque iteraciones que hace, quisiera haber probado esta prueba en un equipo con pocos recursos para poder ver cuanto tiempo demora

La complejidad de la solución es de *O(n*m)* donde n es el tamaño del primer conjunto de datos y m es el tamaño del segundo conjunto de datos, por lo que mientras más caracteres insertemos más tiempo tardara la ejecución, , mientras que el tiempo de ejecución real puede depender de varios factores, como la velocidad del procesador, la memoria disponible y la implementación del algoritmo.

En cuanto implicación ética de un resaltador de sintaxis podemos contemplar el manipular a los usuarios mediante los tonos por lo que no sería ética, sin embargo, podemos pensar que tenemos una gran responsabilidad con la sociedad porque puede ser que un alguien más use nuestro resaltador y este aprendiendo apenas la sintaxis de C#, otra cosa es que al manejar como tal un texto el poder procesar datos personales debe de ser cuidadosamente.

Ricardo Rosales: En este caso el tiempo de ejecución del programa del resaltador de sintaxis varía dependiendo de diversos factores, uno de ellos siendo la capacidad de procesamiento de la computador en la que se ejecuta el programa, ya que en este caso lo estamos midiendo usando la función time de racket, la cual no arroja 3 resultados el tiempo de cpu, el tiempo real y el tiempo de recolección de basura, en una computadora con un mejor procesador el tiempo real de ejecución suele estar entre los 3 y 9 segundos mientras que en una computador con un menor nivel de procesamiento menor, time nos arroja un tiempo real de hasta 44 segundos. Por otro lado otro de los factores que pueden afectar el tiempo de ejecución del programa es el número de líneas y caracteres del archivo fuente de c# que se recibe como input. Y es principalmente aquí donde entra la complejidad espacio temporal de nuestro programa. En este caso nuestro programa cuenta con una complejidad de O(n * m) donde n sería el número de líneas del código fuente mientras que m sería el número de caracteres de cada línea del archivo de entrada.

Por último, en este caso este tipo de tecnología, hablando puramente del resaltador de sintaxis, como tal no tendría un impacto en la sociedad más allá de la comunidad de programadores de c#. Mientras que por otro lado las tecnologías implementadas como la expresiones regulares si suelen tener un impacto más importante en la sociedad, ya que aunque su área de enfoque sigue siendo en área tecnológica o computacional, se aplican expresiones regulares en bastantes herramientas y servicios que a día de hoy son indispensables, un ejemplo de esto es el cómo se usan las expresiones regulares en el campo de la ciberseguridad, desde cosas tan sencillas que usamos dia dia como lo es un filtrador de correos de spam hasta para el hecho de reconocer y detectar patrones de ataques.