*Instituto Tecnológico de Costa Rica*

*Sede Regional San Carlos*

*Escuela Ingeniería en Computación*

*Compiladores e Intérpretes*

***Tarea Programada #1***

***“Mini - C#”***

*Responsable:*

*Hellen Rojas Rojas 2013083934*

*Henry Solís Chacón    2013085706*

*6 de abril del 2016*

*Santa Clara, San Carlos*

**Análisis del lenguaje**

El lenguaje implementado solo posee una parte del lenguaje original c#, pero aun así, tiene implementada una gran cantidad de funciones básicas del lenguajes tales como declaración de variables, asignación, ciclos, clases, métodos, entre otras que permiten la escritura de códigos básicos. Se debe destacar que existen algunas diferencias entre el lenguaje en cuestión y el lenguaje original, por ejemplo luego de declarar una clase, antes de poner corchetes se pueden declarar variables, constantes, y otras clases, además luego de declarar un método antes de los corchetes se pueden declarar variables, esto no es permitido en el lenguaje c# original.

La gramática del lenguaje es muy simple pero cumple con su propósito, que es permitir a los estudiantes entender el proceso del scanner y el parecer.

**Soluciones e implementación**

Para la implementación del proyecto se trabajó la lógica del compilador sobre dos archivos Lexer o Parser, donde el primero contiene la gramática que va a evaluar el Parser

Para la implementación del Lexer y parser se tomó como guía los ejemplos dados en clase por el profesor. Además para la realización de los mismos se tradujo la gramática dada por el profesor, esta contenía todas las reglas excepto la del for each, esta fue creada investigando por otros medios.

Estos dos archivos comprueban la correcta escritura o sintaxis del código recibido,

una vez que se comprueba y se logra compilar se genera un AST que es recorrido e impreso(por medios del método visitor) por un archivo llamado Pretty Print el cual utiliza el componente TreeView para mostrarlo.

Para el manejo de errores se utilizó una clase llamada BaseErrorListener en el archivo parseErrorListener y DefaultErrorStrategy en el archivo DefaultErrorStrategy1 estas clases fueron útiles para capturar los errores sintácticos y poder mostrarlos en la consola de errores en español, ya que se solicitó de esta forma.

En cuanto a la parte visual, el proyecto posee una interfaz amigable con el usuario, intuitiva y fácil de usar, permite la creación de pestañas para trabajar varios archivos al mismo tiempo, posee las funciones básicas en el manejo de archivos tales como guardar, cerrar, copiar, pegar y además el botón de compilar, además posee un cuadro de despliegue de errores y otro donde se muestra el árbol de sintaxis.

**Resultados obtenidos**

|  |  |  |
| --- | --- | --- |
| Objetivo | Estado | Descripción |
| Comentario normal | Completado |  |
| Comentario anidado | Completado |  |
| Caracteres ignorados | Completado |  |
| Token | Completado |  |
| Constantes numéricas, carácter y operadores | Completado |  |
| Error de caracteres invalidados | Completado |  |
| Muestra de información en los errores | Completado |  |
| Conversión del documento adjunto al Parser | Completado |  |
| Opción compilar | Completado |  |
| Opción AST | Completado |  |
| Mostrar errores en editor | Completado |  |
| Manejo de archivos | Completado |  |
| Control de pestañas | Completado |  |
| Identificación de fila y columnas | Completado |  |

**Conclusiones**

El trabajo asignado es una buena manera de entender el funcionamiento de un compilador, dado que muchas veces en los cursos la teoría simplemente se da pero nunca se logra tener una idea práctica o clara del funcionamiento real, es interesante observar la preparación de sus reglas a evaluar hasta su generación del AST.

La herramienta ANTRL4, es algo compleja de entender por su extraña sintaxis y por los varios problemas que presentar al intentar aplicar mejoras o cambios a una sección, sin embargo su facilidad de ayuda para generar el compilador es muy buena.

Una de las mayores dificultades es lograr obtener un parser capaz de reconocer todos los errores del código, para evitar fallos en la futura sección de evaluación contextual.

# Bibliografía

ANTRL4. (05 de 04 de 2016). Obtenido de http://www.antlr.org/api/JavaTool/org/antlr/v4/runtime/DefaultErrorStrategy.html#endErrorCondition(org.antlr.v4.runtime.Parser)

Microsoft Development Network. (05 de 04 de 2016). *Microsoft Development Network*. Obtenido de https://msdn.microsoft.com/es-es/library/aa664754(v=vs.71).aspx

Universidad de Cordoba. (05 de 04 de 2016). *UCO*. Obtenido de http://www.uco.es/users/ma1fegan/Comunes/manuales/pl/ANTLR/Analisis-lexico-con-ANTLR.pdf