Instituto Tecnológico de Costa Rica
Sede Regional San Carlos

Escuela de Computación

Compiladores e intérpretes

Proyecto 1

Responsables:
Crisly González Sánchez
Kevin Zamora

18 de Septiembre del 2016

Santa Clara, San Carlos

Introducción

Un compilador es un programa que tiene la funcionalidad de traducir el código fuente escrito en un lenguaje de alto nivel de un sistema en ejecución a lenguaje máquina, para que pueda ser procesado por el ordenador.

Como partes de un compilador tenemos el backend y frontend, la principal función del frontend es la interacción con el cliente, además verifica que el código fuente sea válido, así como también genera un árbol de derivación y a su misma vez rellena la tabla de símbolos. Y a nivel de backend como hemos mencionado anteriormente el compilador en esta etapa se encarga de traducir el código fuente a lenguaje máquina mediante procesos establecidos.

Es importante mencionar los tipos de compiladores que existen en los cuales son: compiladores cruzados, compiladores optimizadores, compiladores de una sola pasada y de varias pasadas, compiladores JIT y los conocidos intérpretes, que traducen el código del programa en tiempo real. Cabe destacar que los compiladores son importantes, ya que mediante ellos hacemos la comunicación desde nuestro software al lenguaje máquina, de no existir no se podría contar con aplicaciones informáticas, ya que son la base de todo sistema.

El principal objetivo de la elaboración de esta documentación, la cual está basada en el primer proyecto programado del curso Compiladores e Intérpretes es el uso de la gramática de mini java, mediante la creación de un archivo con tokens, llamado LexerMiniJava y otro denominado ParserMiniLexer, los cuales se harán uso mediante un editor de texto que probará código fuente y mediante el compilador reportará los errores causados en la prueba.

Se hizo uso de la herramienta ANTLR la cual es una librería que permite la creación de compiladores o intérpretes en distintos lenguajes PHP.SQL,JAVA.

Análisis del problema

Deberá crearse un compilador basado en la gramática del lenguaje de programación Java por medio de la herramienta ANTLR 4, es requerido el análisis de la gramática de mini Java proporcionada previamente por el profesor en el enunciado del proyecto programado. Es deseable el uso de IDE, intelliJ de JetBrains para el uso de la herramienta ANTLR la cual es de sencilla integración, el uso de la librería mencionada anteriormente cuenta con la documentación necesaria para la investigación del equipo de desarrollo, lo cual marcará la pauta en el avance del proyecto.

El compilador necesitará la creación de un archivo para contener los tokens de la gramática el cual deberá ser denominado como LexerMiniJava, además un archivo con las reglas de la gramática previamente asignada, este tendrá denominación de ParserMiniJava. Mediante el análisis cuidadoso de la gramática los programadores deberán identificar los tokens los cuales son representados mediante “isTokens”.

Se ha solicitado que el compilador cuente con una interfaz que permita el ingreso del código prueba del compilador, un botón de compilar, cargar un archivo y guardarlo con extensión “.java”. La interfaz deberá contar con dichas funciones principales, además de tener una interfaz intuitiva al usuario.

Además deberá hacer un reporte de los errores que posea el código prueba del usuario, con respecto a las reglas establecidas por el programador en el Scanner, es importante tomar en cuenta que los errores deberán ser mostrados en una pantalla inferior de la herramienta, por lo cual se recomienda usar un textArea que permita dicha visualización. La herramienta ANTLR tiene 4 tipos de errores principales que deberán ser programados en una clase de errores, los cuales son : NoViableAltException , LexerNoViableAltException , FailedPredicateException, InputMismatchException; deberá de realizarse una investigación de sus usos e implementación para la creación de la clase errores en el scanner.

Es importante tomar en cuenta que los errores deberán tener una traducción a lenguaje español, mostrando la fila y columna donde sucedió el error, de esta manera el usuario podrá percatarse del problema y lo podrá resolver secillamente.

El compilador requiere que se realice la implementación de un árbol AST (Árbol Abstracto de Sintáxis) el cual será usado en las fases posteriores a la entrega, para el desarrollo de la asignación deberá estar creado y desplegarse con sus respectivas etiquetas en cada sub-árbol. La impresión del árbol se hará en consola para la respectiva etapa, pero deberá dejarse programado para que sea escalable.

Solución del problema

El equipo de desarrollo hizo uso de la librería ANTLR versión 4 para realizar el compilador solicitado con la la gramática de miniJava, previamente asignada en el enunciado de proyecto por el profesor. Además se utilizó el IDE IntelliJ como era deseable mediante los requerimientos entregados.

El compilador cuenta con una clase LexerMiniJava.g4 (extensión asignada a los archivos ANTLR) la cual contiene todos los tokens encontrados en la interpretación y lectura de la gramática, fueron identificados mediante las “ ” , que comúnmente que facilitan la interpretación de los tokens en las reglas de la gramática. Los tokens fueron seleccionados y creados en el archivo mediante la declaración nombreToken: “token” ; (representación simple). La herramienta tiene por definición que los tokens son identificados en los archivos.g4 porque inician con mayúscula mientras que las reglas gramaticales deberán ser escritas en minúscula, aspectos importante a tomar en cuenta al trabajar en la librería.
La investigación sobre la herramienta ANTLR y la lectura del manual de usuario para su uso ayudo a la redacción de los tokens mediante el lenguaje Java, para que estuvieran sintácticamente correctos. No estará de más mencionar que en el archivo es requerido colocar lexer grammar LexerMiniJava; en la parte superior del mismo, esto porque al ser archivos independientes tanto Lexer como Parser deberá identificarse en cual se está trabajando.

También se creó un archivo extensión .g4 para el Parser o Scanner el cual contiene las reglas sintácticas de la gramática de miniJava, además de hacer uso de los tokens del LexerMiniJava para la construcción de las reglas, cabe mencionar que dichas reglas han sido comprobadas con las declaraciones correctas en lenguaje Java, esto con el fin de comprender la gramática y hacer una transcripción de sus reglas correctamente.

El compilador cuenta con una clase de reporte de errores en los cuales se obtienen los errores que generalmente son mostrados en consola y se traducen a lenguaje

español para luego ser proyectados en textArea en la interfaz de editor. Se hizo uso de los 4 tipos de excepciones comunes en la librería ANTLR de la cual, cualquiera de las que sea necesario activar llevarán al diagnóstico del error como fue solicitado, mediante interfaz, además indicará la fila y columna de donde sucedió el problema.

El editor de texto utilizado fue obtenido de internet y modificado para que se adaptara a las necesidades del scanner que se requería, donde debía tener un botón de compilar, guardar archivos con extensión “.java” , manejo de funciones básica de archivo y un apartado de errores, además de que muestra la fila y columna donde esta posicionado el usuario.

Y finalmente se realizó la creación de un árbol AST, de acuerdo a la gramática para su estructura, en la cual se realizó una clase imprimirAST donde están los métodos visit de cada estructura del árbol hasta llegar a sus hojas e imprimir los resultados.

Análisis de resultados

Tarea	Estado	Observaciones
Creación del editor de texto con los manejos básico del archivo, implementación de extensió “ . java “	100%	
Compilación texto desde la interfaz	100%	
Reportes de errores con línea, columna y traducción de error al español en interfaz	100%	
Manejo de excepciones básicas de la herramienta ANTLR	100%	
Creación de archivo Lexer, con reglas gramaticales de Java, identificación de tokens en gramática MiniJava	100%	
Creación de reglas gramaticales de MiniJava en archivo Parse	100%	
Elaboración de casos de prueba a la archivo ParseMiniJava	100%	
Creación del árbol AST	100%	
Visualización del árbol AST para las siguientes etapas del proyecto	100%	

Conclusión

El desarrollo del proyecto asignado tuvo como base la comprensión e investigación de la gramática de MiniJava, para lograr la creación de reglas sintácticas y su validación.

Se logró comprender el proceso que siguen los compiladores y su interpretación de código de alto nivel, la traducción que realiza y cómo es su manejo internamente. Caso que usualmente no prestamos atención mientras programamos.

La extensión de la gramática utilizada permitió lograr el alcance propuesto por el equipo de desarrollo a partir de los requerimientos del sistema.

Recomendaciones

- Es recomendable realizar la impresión del árbol en un apartado del editor de texto para una mejor visualización de la estructura de las clases. Además de que dicha impresión sea atractiva y comprensiva a la lectura del usuario.

- El sistema de errores podrá mejorarse a una mayor cobertura de problemas que la clase actual no cubre, de la cual se pueden desarrollar casos de prueba que verifiquen los escenarios posibles de fallo y sean corregidos antes de la creación del árbol.

- Es recomendable comprender la gramática de MiniJava antes de realizar cualquier corrección en el archivo de ParserMiniJava y LexerMiniJava, creados para el funcionamiento del compilador.

- Es deseable que los errores puedan traducirse a 2 lenguajes aún habiéndose mostrado en pantalla, mediante una actualización de boton en la ventana de textArea.

Bibliografía

SR. (SR). ¿Qué es un compilador?. 16 de Septiembre, de culturacion Sitio web: http://culturacion.com/que-es-un-compilador/

SAMUEL PADILLA. (SR). Proceso de Compilación. 16 de Septiembre, de javacurso Sitio web: https://javacurso.wordpress.com/compilar-java/

Terence Parr. (2013-01-15). The Definitive ANTLR 4 Reference. 16 de Septiembre, de ANTLR Sitio web: https://pragprog.com/book/tpantlr2/the-definitive-antlr-4-reference