Instituto Tecnológico de Costa Rica

Compiladores e Intérpretes

Primer proyecto:

Análisis léxico y sintáctico

Profesor:

Allan Rodríguez Dávila

Estudiantes:

Bryan Londoño Marchena Dylan

Rodríguez Chavarría Segundo

Semestre 2025

Fecha de entrega: 6/10/2025

Manual de usuario y pruebas de funcionalidad: Para este apartado se optó por realizar un video en el cual se explican y muestran las características del proyecto a través de distintas pruebas y archivos.

Link del video: https://youtu.be/pq7Wj8QmP-0

Descripción del problema: El problema trata sobre un grupo de desarrolladores los cuales quieren crear un nuevo lenguaje imperativo para crear chips pero a su vez este debe ser potente y rápido, para esto será necesario que se utilicen herramientas como lo son Jflex y Cup con los cuales se podrán crear tanto un scanner como un parser los cuales son necesarios para la primera fase del compilador, ya que de alguna manera estos se encargaran de revisar primeramente lo que es sintaxis y gramática.

Diseño del programa: En general el programa se empezó con el cup ya que queríamos saber si la gramática que habíamos realizado en la tarea 1 seguiria con errores o no, por lo cual eso fue nuestro paso inicial, la cual claramente representó problemas de ambigüedad, sin embargo se corrigieron y se procedió con la creación del lexer el cual ya generaba los tokens y lexemas para que así se pudieran pasar al cup y revisar si la gramática está funcionando de la forma esperada o no, una vez realizadas todas estas correcciones se hicieron los errores en modo pánico tanto para el lexer como el cup, además de esto se hizo la creación de un archivo fuente para probar y algo que es de suma importancia fue el uso de netbeans para usarlo como un ambiente en el cual automatizamos para la compilación, ademá para los algoritmos fueron utilizados tanto derivación por izquierda, derivación por derecha y recursividad.

Librerías utilizadas:

import java_cup.runtime.* comunicación entre parser y lexer,

import java.io.* esta es para entrada y salida,

import java.util.ArrayList para utilizar listas;

Análisis de resultados:

Requerimiento	¿Funciona?
El sistema debe leer un archivo fuente.	Sí
Se debe escribir en un archivo todos los tokens encontrados, identificador asociado con el lexema.	Sí
Por cada token deberán indicar en cuál tabla de símbolos va y cual información se almacenará.	Sí
Indicar si el archivo fuente puede o no ser generado por la gramática.	Sí
Reportar y manejar los errores léxicos y sintácticos encontrados. Debe utilizar la técnica de Recuperación en Modo Pánico	Sí
Indica la línea del error	Sí
¿El Scanner funciona correctamente?	Sí
¿El Parser funciona correctamente?	Sí
¿Se mantiene la intención original de la gramática?	Sí

Justificación:

Durante la realización del proyecto uno de los principales inconvenientes fueron las operaciones lógicas, relacionales y aritméticas. Se decidió hacerlo de forma jerárquica donde se comenzara en el orden mencionado anteriormente y se fuera descendiendo hasta llegar a poder insertar los operadores. Esta fue la forma en la que se decidió hacer debido a que se respetaba el orden de precedencia donde se encuentran primero las operaciones que se llevan a cabo de último y si se lleva a cabo de forma manual la derivación a partir de la producción logical_operator_and se pueden realizar combinaciones de las operaciones lógicas, relacionales y aritméticas. Como tal esta estructura compromete el cumplimiento de ciertos lineamientos establecidos para el proyecto, ya que se permitiría el reconocimiento de ciertas combinaciones donde se operen elementos que no podrían, tales como un booleano y un entero, por mencionar un ejemplo. Además, esta fue la forma en la que se pudo desarrollar sin que se reportaran ambigüedades.
Se establecieron en el cup algunas reglas de precedencia donde para algunas aritméticas (suma, resta, división, multiplicación) , relacionales (con la excepción del not) se definieron reglas de izquierda. Mientras que para las unarias, not y la potencia la precedencia es por la derecha. De forma general la recursión en el resto de producciones se dan por la izquierda, como por ejemplo en los parámetros de un función.
Para mayor facilidad se definió que el programa puede estar constituido por variables globales, el programa principal y las funciones. De estos solo el programa principal es necesario y deben de estar en este orden debido a que se definió de esta forma para no entorpecer otros procesos

Bitácora: Bryan9044/ProyectoCompi-1

Referencias:

Emerge Tu Mundo: Aura y lo Errores de CUP