Desarrollo de Aplicaciones Avanzadas (Grupo 501)

Mini Proyecto - Baby Duck

Profesores:

Dr. Jesús Guillermo Falcón Cardona
Ing. Elda G. Quiroga
Dr. Iván Mauricio Amaya Contreras

Jose Alberto Kaun Sada
A01720829

Introducción

Un compilador es capaz de convertir el código fuente que se ha escrito en un lenguaje de programación en un lenguaje de máquina, lo que facilita la ejecución del código en una computadora.

El objetivo de este proyecto es desarrollar un compilado. El análisis léxico, el análisis sintáctico, el análisis semántico, la generación de código intermedio, la optimización del código y la generación de código de máquina son componentes esenciales de un compilador. El compilador se construyó en detalle en este reporte.

Análisis Léxico

Definición de Tokens: Los tokens definidos son:

- Palabras clave (PROGRAM, VAR, BEGIN, END, etc.)
- Tipos (STRING_CONST, BOOLEAN, INT, REAL, CHAR, etc.)
- Operadores (aritméticos como PLUS, MINUS, lógicos como AND, OR, etc.)
- Símbolos de sintaxis (LPAR, RPAR, LBRAC, RBRAC, etc.)
- Repetidores (FOR, WHILE,DO)
- Condicionales(IF, ELSE, THEN, etc.)

Expresiones Regulares

Cada token está asociado con una expresión regular que define su patrón en el código fuente. Por ejemplo, t_PLUS = r'\+' identifica el operador de suma.

Tokens Específicos
Algunos tokens, como IDENTIFIER y NUMBER_CONST, necesitan funciones únicas para ser identificados y procesados. Las expresiones regulares se utilizan en estas funciones para identificar patrones y asignar el tipo de token adecuado.

Analisis del codigo fuente
Finalmente, el lexer demuestra su habilidad para procesar archivos reales leyendo un archivo de entrada, tokenizando(?) su contenido y escribiendo los resultados en un archivo de salida.

Análisis Sintáctico

El siguiente paso importante en un compilador es el análisis sintáctico, que generalmente se lleva a cabo por un componente llamado parser. Es fundamental comprender su función y cómo habría funcionado con el analizador léxico.

Construcción del Árbol de Análisis Sintáctico(Parse tree)
El analizador léxico crea tokens, que el parser organiza en una estructura de árbol que muestra la estructura sintáctica del código fuente. Esto es importante para la fase de compilación.

Uso de Gramática
El parser se basa en una gramática formal que establece cómo se pueden combinar los tokens para formar construcciones sintácticas aceptables. Esta gramática es una colección de reglas que definen la sintaxis del lenguaje.

Detección y Manejo de Errores Sintácticos
El parser es responsable de detectar errores sintácticos además de construir el Parse Tree. Si encuentra una secuencia de tokens que no se ajusta a la gramática, genera un error. Por ejemplo falta de símbolos, puntuacion, uso incorrecto de operadores, etc.

Clasificación del Código

El código está realizado en Python utilizando la biblioteca PLY (Python Lex-Yacc). Este tipo de código es un componente del procesamiento de lenguajes de programación y está destinado al análisis léxico, análisis sintáctico, VMs y generaciones de código. Algunas características son:

Flexibilidad y adaptabilidad: la capacidad de crear y cambiar tokens según las necesidades de un lenguaje de programación específico.
Robustez en el Manejo de Errores: La implementación incluye la gestión efectiva de errores léxicos, que es esencial para un análisis léxico confiable.
Interacción con las etapas adicionales del compilador: diseñado para trabajar bien con las fases de compilación posteriores, como el análisis sintáctico.

Conclusion

El desarrollo de un compilador para el micro-lenguaje "baby_duck" se ha centrado en la fase de análisis léxico y ha teorizado sobre las fases de análisis sintáctico. El código del lexer, un ejemplo evidente de análisis léxico en Python con PLY, muestra la capacidad de desglosar eficazmente un programa en tokens, un paso esencial en cualquier proceso de compilación.

La discusión teórica proporciona una base sólida para entender estos componentes, aunque la implementación del parser y el cubo semántico aún no está completa. El proyecto enfatiza que una estructura bien definida y una planificación en la construcción de compiladores y un manejo de errores en todas las etapas son esenciales.