Implementación del analizador léxico y sintáctico para el compilador del lenguaje “Phyton”

Integrantes:

Frederick Antonio Balderrama Mendoza
e-mail: antoniobalderramamendoza@gmail.com
Samuel Lucas Gossweiler Rivas
e-mail: caleb.antonio88823@gmail.com
Horst Andrés Gottlieb Guzmán
e-mail: andres.gottlieb10@gmail.com
Frederick Antonio Balderrama Mendoza
e-mail: antoniobalderramamendoza@gmail.com
Adrián Montaño Ferrufino
e-mail: adrianmontano1@upb.edu
Gabriel Alejando Villarreal Ponce
e-mail: gabriel.villarreal.ponce@gmail.com
Diego Ibáñez Caero
e-mail: diegoibanez912@gmail.com

Cochabamba-Bolivia
29/05/2024

Índice

Resumen:

En este documento, se presenta la explicación del proceso de creación del analizador léxico y sintáctico del compilador de un nuevo lenguaje de programación llamado Phyton. Se da a conocer los elementos requeridos para el proceso, y se muestra cada etapa del proceso que tienen elementos esenciales para la creación de ambos analizadores.

1. Introducción

El problema planteado al principio de la materia es el de poder crear un compilador desde 0 donde se pueda definir un lenguaje de programación nuevo, el cual sea traducido y entendido por este compilador. Como bien se sabe, crear un compilador desde 0 es un proceso bastante tedioso. Por esta razón, se decidió en primera instancia implementar el analizador léxico y sintáctico del compilador. Este compilador funciona y traduce el lenguaje de programación Phyton.

El lenguaje de programación Python, fue creado a partir del idioma latín, el cual se puede apreciar en el nombre de las distintas palabras reservadas y funcionalidades implementadas del lenguaje. El lenguaje en sí tendrá como objetivo principal demostrar todo el proceso de implementación de un compilador desde cero, así como todo el desarrollo de un lenguaje formal y el proceso de creación del analizador léxico y sintáctico de este compilador.

Como se dijo anteriormente el objetivo de este documento es el de mostrar el proceso de diseño del analizador léxico y sintáctico del compilador de Phyton, el cual deberá interpretar de manera correcta el lenguaje Phyton y será capaz de procesar y estructurar código fuente de acuerdo a las especificaciones del lenguaje. Por otra parte, un objetivo importante es poder demostrar que las implementaciones desarrolladas permiten sentar las bases para poder desarrollar un compilador robusto y extensible.

También se tiene como objetivos más específicos, definir las reglas del lenguaje de manera clara y entendible, para poder crear e integrar el analizador léxico y sintáctico sin fallas.

2. Delimitación

Hasta el momento, se ha implementado de manera satisfactoria el analizador léxico y el analizador sintáctico del compilador del lenguaje de programación Python. Estas dos componentes esenciales permiten identificar y categorizar correctamente los tokens del código fuente, así como estructurar estos tokens según las reglas gramaticales del lenguaje. No obstante, el proyecto presenta ciertas limitaciones que deben ser consideradas.

En primer lugar, no se ha desarrollado el analizador semántico, que es crucial para verificar la coherencia y validez de las estructuras sintácticas en términos de reglas de lógica y semántica del lenguaje. También falta la generación de código intermedio, como código assembler, que actúa como intermediario entre el código fuente de alto nivel y el

código máquina. Además, aún no se ha implementado la generación de código máquina, lo cual es necesario para que el compilador pueda producir programas ejecutables directamente en hardware.

Debido a la ausencia de estos componentes, el compilador no está completamente funcional. En su estado actual, no puede producir programas ejecutables ni realizar una verificación semántica exhaustiva del código fuente. Esta funcionalidad limitada es una de las principales áreas que necesitan desarrollo futuro.

Se tiene en mente mejorar y optimizar los procesos del analizador léxico y sintáctico actuales. Esto incluye la optimización del rendimiento y la eficiencia, así como la reducción de posibles errores o ineficiencias. En futuros avances, se podrá medir la eficiencia del compilador, evaluando aspectos como el tiempo de procesamiento, el uso de memoria y la precisión en la detección de errores.

Actualmente, el compilador devuelve el Árbol de Sintaxis Abstracta (AST), que representa la jerarquía gramatical del programa. Este es un paso crucial, ya que el AST es la base para las fases posteriores de análisis y generación de código. Sin embargo, la producción del AST es solo una parte del proceso completo de compilación.

Estas delimitaciones indican que, aunque se ha logrado un progreso significativo en la implementación del analizador léxico y sintáctico, el proyecto aún está en una fase inicial. Los futuros desarrollos se enfocarán en completar las partes faltantes del compilador y en optimizar los componentes ya implementados, para lograr un compilador totalmente funcional y eficiente.

3. Metodología

Para desarrollar el compilador desde 0, utilizamos una metodología colaborativa en la que integramos herramientas modernas y eficientes, empleamos git para la gestión del código, lo que nos permitió llevar un control preciso de las versiones y facilitar la integración de los aportes de cada miembro del equipo. Las reuniones regulares en clases permitieron la correcta coordinación del equipo para la resolución de problemas, además utilizamos la funcionalidad de colaboración en tiempo real de Phyton, lo que permitió trabajar de manera simultánea en el mismo código desde diferentes computadoras.

En primer lugar, se comenzó el proyecto definiendo las palabras reservadas y reglas del lenguaje Phyton, para después empezar a definir reglas de producciones para poder verificar la validez sintáctica de nuestras declaraciones de código. Esto fue lo que más tiempo tomo en términos de pensar para no crear un lenguaje ambiguo.

Con el trabajo colaborativo se logró juntar diversas ideas de implementaciones y de esta forma poner en práctica todo el conocimiento adquirido en la materia hasta el momento. Implementando el parser y lexer se reforzó aún más los conceptos que se aprendieron y ayudó a entender aún más a fondo los fundamentos de un compilador. También se trabajó de manera conjunta en la creación de un nuevo lenguaje de programación que fue un proceso altamente creativo y divertido.

4. Conclusiones

Se ha logrado desarrollar el analizador léxico y sintáctico de un compilador para interpretar Python, marcando un hito significativo en el proyecto. Este logro no solo ha permitido consolidar los conocimientos previos, sino también aplicar nuevas habilidades adquiridas durante el curso. La elección de Python como lenguaje base ha facilitado enormemente la implementación del lexer y parser, gracias a la claridad y solidez de sus reglas gramaticales.

Sin embargo, es importante señalar que el compilador aún presenta limitaciones significativas. Aunque se han completado las fases de análisis léxico y sintáctico, aún falta implementar componentes cruciales como el analizador semántico y la generación de código intermedio y máquina. Esta ausencia impide que el compilador esté totalmente funcional en su estado actual, limitando su capacidad para producir programas ejecutables y realizar una verificación semántica exhaustiva.

Además, se identifica la necesidad de mejorar y optimizar los procesos del analizador léxico y sintáctico existentes. Estos esfuerzos se centrarán en la optimización del rendimiento y la eficiencia, así como en la reducción de posibles errores o ineficiencias. A medida que avancemos en el desarrollo, se realizarán mediciones de eficiencia para evaluar el rendimiento del compilador y su capacidad para procesar código de manera efectiva.

A pesar de estas limitaciones, el logro inicial de implementar con éxito los analizadores léxico y sintáctico sienta una base sólida para el desarrollo continuo del proyecto. La elección de Python como lenguaje base y el enfoque claro y estructurado en la implementación de cada etapa del proyecto son elementos clave que nos permitirán abordar con éxito los desafíos futuros y avanzar hacia la finalización del compilador de Phyton.

5. Recomendaciones

Algunas recomendaciones para investigaciones futuras en el tema de analisis lexico y sintactico en compiladores(de un lenguaje nuevo) podrían ser:

- **Integración con herramientas de desarrollo modernas:** explorar la integración del compilador con entornos de desarrollo integrados (IDEs) modernos, proporcionando características como resaltado de sintaxis en tiempo real, autocompletado y depuración interactiva.

- **Refactorización de código:** Implementar herramientas que faciliten la refactorización automática del código, como la renombrados de variables y funciones, la extracción de métodos y la optimización del código redundante.

- **Asistente de codificación basado en IA:** Explorar el uso de asistentes de codificación basados en inteligencia artificial que puedan sugerir mejoras, corregir errores y ofrecer recomendaciones de diseño basadas en el análisis del código.

- **Integración de análisis de rendimiento:** Utilizar algoritmos de IA para analizar el rendimiento del código en tiempo real. Estos algoritmos pueden identificar patrones y detectar automáticamente secciones del código que causan cuellos de botella, como bucles ineficientes o llamadas a funciones costosas.