CC3032 - Construcción de Compiladores
Proyecto 1: Análisis Semántico

Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencia de la Computación

Gabriel Brolo, Bidkar Pojoy

Ciclo 2, 2024

Introducción

En este primer laboratorio, deberán implementar el análisis semántico para un lenguaje de programación llamado CompiScript. Ustedes cuentan con otro documento de referencia que deben consultar para aprender sobre la sintaxis y otras generalidades del lenguaje.

- Por favor, consultar el documento "Generalidades de CompiScript".

Requerimientos

Tareas Directas

1. Crear un analizador sintáctico utilizando ANTLR o cualquier otra herramienta similar de su elección.

   a) Se recomienda utilizar ANTLR, pero usted es el amo y señor, la ama y señora, de sus decisiones de diseño y arquitectura.

   b) Dedique tiempo a pensar la arquitectura de su compilador y el ambiente en general.

2. Añadir acciones semánticas en este analizador sintáctico y construir un árbol sintáctico.

3. Implementar la recorrida de este árbol utilizando ANTLR Listeners o Visitors para evaluar las reglas semánticas que se ajusten al lenguaje.

4. Construir una tabla de símbolos que interactúe con cada fase de la compilación, incluyendo las fases mencionadas anteriormente. Esta tabla debe considerar el manejo de entornos y almacenar toda la información necesaria para esta y futuras fases de compilación.

IDE Interactivo

1. Desarrollar un IDE, es decir, una interfaz de desarrollo interactiva que permita a los usuarios escribir su propio código y luego compilarlo, siguiendo todos los requerimientos anteriores.

2. Para este propósito, se permite utilizar cualquier framework de desarrollo web y crear una API para realizar llamadas a sus compiladores, o hacerlo de cualquier otra manera que se considere adecuada.

a) Es decir, puede crear una arquitectura de microservicios o bien un monolito.

b) Usted elija la arquitectura que considere más apropiada, lo importante es tener el IDE visual para la ejecución de sus códigos de CompiScript.

Sugerencias de Frameworks y Arquitecturas

A continuación se ofrecen algunas sugerencias de frameworks y arquitecturas que pueden ser útiles para la implementación del compilador y el IDE:

- **Frameworks de Desarrollo Web:**
  - *React*: Para construir interfaces de usuario interactivas.
  - *Vue.js*: Otra opción popular para la construcción de interfaces de usuario.
  - *Angular*: Un framework robusto para aplicaciones web.
  - *Django*: Un framework de Python robusto para aplicaciones web.
  - *Flask*: Un framework de Python robusto para aplicaciones web.
  - *Vaadin*: Un framework de Java robusto para aplicaciones web.
- **Librerías, lenguajes y Herramientas:**
  - *ANTLR*: Para la generación del analizador sintáctico.
  - *Node.js*: Para la creación de la API del compilador.
  - *Express*: Un framework para Node.js que facilita la creación de APIs.
  - *Java, Go, C++*: Un lenguaje tipado ayuda bastante en la creación de un compilador porque da una buena estructura sobre la implementación. Si quieren integrarse después a herramientas como ANTLR no necesitan un SDK como tal, pueden usar el jar de ANTLR (Java) para generar sus clases necesarias de recorrido e implementar otros aspectos del diseño con otros lenguajes. También pueden usar las librerías de Python o Java de ANTLR directamente. O bien otra tool que estén usando.
- **Arquitecturas:**
  - Arquitectura basada en microservicios: Donde cada componente del compilador se implementa como un servicio independiente que puede ser desplegado y escalado de forma autónoma.
  - Arquitectura de cliente-servidor: Donde el IDE funciona como un cliente que envía el código fuente a un servidor para su compilación y luego recibe el resultado.
  - Monolito: Sería lo más similar a un compilador convencional en el aspecto que aumenta la portabilidad al no tener demasiados componentes intermedios. Ustedes mandan, lo importante es que tenga una arquitectura eficiente.

Consideraciones Semánticas

Basado en el documento de CompiScript, algunas de las reglas semánticas que deben ser implementadas incluyen, pero no se limitan a:

- **Sistema de Tipos:**
  - Verificación de tipos en operaciones aritméticas y lógicas.
  - Control de tipos en asignaciones y declaraciones de variables.

- Compatibilidad de tipos en expresiones de comparación.

- **Manejo de Ámbito:**
  - Correcta resolución de nombres de variables y funciones según el ámbito.
  - Detección de variables no declaradas.
  - Control de acceso a variables globales y locales.
- **Funciones y Procedimientos:**
  - Verificación de la cantidad y tipo de argumentos en llamadas a funciones.
  - Validación del tipo de retorno en funciones.
- **Control de Flujo:**
  - Asegurar que las condiciones de los ciclos y estructuras condicionales sean de tipo booleano.

Entrega

La entrega de este laboratorio debe incluir:

1. El código fuente del analizador sintáctico y semántico.

   a) Entrega a través de Canvas por medio de un documento PDF con el reporte de su proyecto y en este documento, un enlace a su repositorio **privado** de GitHub que deben compartir con su catedrático y auxiliares. Revisen la página de Canvas con los datos de los usuarios de GitHub.

2. La implementación del IDE interactivo.
3. Documentación que explique cómo compilar y ejecutar el proyecto, dentro del reporte mencionado anteriormente.
4. Documentación que explique a detalle la arquitectura de su compilador e IDE, dentro del mismo reporte mencionado anteriormente, que servirá a los calificadores para comprender sus decisiones de diseño y analizar la veracidad de su implementación.
5. Ejemplos de código en CompiScript y su correspondiente análisis semántico.
6. Un video no listado en YouTube con la ejecución de la compilación hasta esta fase de no más de 5 minutos de duración.

Ponderación

Este laboratorio tiene como objetivo principal que ustedes comprendan y apliquen técnicas de análisis semántico en el contexto de la construcción de compiladores, desarrollando habilidades prácticas en la implementación de un lenguaje de programación y su entorno de desarrollo. La ponderación es como sigue:

Componente	**Puntos**
IDE	15 pts
Analizador Sintáctico y Semántico con reglas y análisis semántico	30 pts
Tabla de Símbolos	25 pts
Ejecución Correcta del Código CompiScript	30 pts
Total	**100 pts**

Cuadro 1: Ponderación de los componentes del proyecto