Universidad Católica del Norte

ESCUELA DE INGENIERÍA UCN COQUIMBO

Analizador Sintáctico

Integrantes:

Manuel Jerez, Nicolás Rojas

17 de junio, 2025.

Introducción

En el desarrollo de lenguajes de programación, es esencial una correcta administración del scope **(alcance de variables)** para generar un código predecible y a largo plazo. Este trabajo expone la creación y puesta en marcha de un compilador con scope por bloques que integra funciones inspiradas en lenguajes consolidados como Java, C y C++.

Contexto y Motivación

Los lenguajes de programación tradicionales manejan el scope de variables de diferentes maneras: scope dinámico (donde las variables se resuelven según el orden de ejecución) versus scope estático o léxico (donde las variables se resuelven según su ubicación en el código fuente). Nuestro lenguaje implementa scope estático por bloques, proporcionando un comportamiento predecible y familiar para desarrolladores.

Desarrollo del lenguaje

Características Principales del Lenguaje

El compilador desarrollado soporta las siguientes funcionalidades esenciales:

- Sistema de Tipos
- **dinoint** : Números enteros *(equivalente a int en Java)*
- **float**: Números decimales de punto flotante
- **paputexto** : Cadenas de texto *(equivalente a String en Java)*
- **Arrays**: Soporte para arreglos de todos los tipos básicos
- Operaciones Aritméticas
- **Operaciones básicas**: suma (+), resta (-), multiplicación (*), división (/)
- **Operaciones mixtas:** Permite operaciones entre enteros y flotantes
- Protección contra división por cero
- Operadores de Comparación
- **Igualdad**: == *(igual)*, != *(diferente)*
- **Relacionales**: > *(mayor)*, < *(menor)*, >= *(mayor o igual)*, <= *(menor o igual)*

Estructuras de Control

- **Condicionales**: si (condición) { } sino { } (equivalente a if/else)
- **Bucles**: mientras (condición) { } (equivalente a while)
- **Bloques independientes:** { } para crear scope local
- Entrada y Salida
- **Output**: texto variable_o_literal para imprimir en pantalla
- **Input:** get variable para leer desde teclado
- **Concatenación:** Operador + para unir cadenas de texto
- Arquitectura del Compilador

El compilador utiliza las herramientas estándar de construcción de compiladores:

- **Flex:** Analizador léxico para reconocimiento de tokens
- **Bison:** Analizador sintáctico para procesamiento de gramática
- **GCC:** Compilador C para generar el ejecutable final
- Docker: "Containerización" para portabilidad y facilidad de despliegue *(para contar con el contorno Linux)*

Implementación de Scope por Bloques

La característica más destacada del compilador es el sistema de scope estático por bloques, implementado mediante:

- Stack de Ambientes: Cada bloque {} crea un nuevo nivel de scope
- Búsqueda Jerárquica: Las variables se buscan desde el scope actual hacia los scopes padre
- Shadowing: Variables locales pueden ocultar variables de scopes superiores
- Gestión Automática de Memoria: Liberación automática al salir de cada scope

Objetivos

Este compilador fue desarrollado con los siguientes objetivos:

- Demostrar la implementación práctica de scope estático en un compilador funcional
- Proporcionar un lenguaje simple pero completo para aplicaciones básicas
- Facilitar el aprendizaje de conceptos de compiladores y scope
- Ofrecer una base sólida para futuras extensiones del lenguaje
- Alcance del Informe
- En las siguientes secciones se detallará:
- El análisis léxico y sintáctico implementado
- La arquitectura interna del sistema de scope
- Ejemplos prácticos de uso del lenguaje
- Pruebas de funcionalidad y validación
- Comparación con otros sistemas de scope
- Conclusiones y trabajo futuro
- Tener buena nota

Conclusión

A lo largo de este taller hemos aprendido a implementar herramientas de análisis léxico y sintáctico tales como, Bison y Flex. Donde a su vez hemos optado por utilizar un Scope estático por bloques para darle mayor funcionalidad al laboratorio. El compilador cuenta con el uso de datos primitivos (int, float, String) los cuales nos permiten generar operaciones básicas, así como sea imprimir por pantalla y hacer cálculos cumpliendo la funcionalidad de una calculadora básica.