Instituto Tecnológico de Costa rica

Escuela de Ingeniería en Computación

Compiladores e Intérpretes [IC - 5701]

Tarea

BNF

Estudiantes:

Yosimar Montenegro Montenegro 2023147365

Eduardo Rojas Gómez 2023152927

Profesor(a) a cargo:

Allan Rodríguez Davila

II Semestre | 2025

Descripción del Problema

Se requiere definir una gramática para un lenguaje de tipado fuerte y explícito con paradigma imperativo. Para ello se utilizan referencias de lenguajes como: C, ADA, Pascal…, se busca permitir una sintaxis completa pero no tan flexible como Python o Javascript. Para la definición de la gramática libre de contexto, se utilizará la notación BNF junto a algunos símbolos de REGEX permitidos por el profesor.

Diseño del Programa

Lista de Terminales:

Palabras reservadas:
let
int
float
boolean
char
string
else
for
return
break
output
input
global
void
principal
decide
of
end
loop
exit
when
to
downto
do.

Operadores: +, -, *, /, //, %, ^, @, ~, Σ, <, >, <=, >=, ==, !=, =, ++, –

Símbolos especiales: $, ¿, ?, |, ¡, !, ,, ;, (,), [,], є, э, ->, \n, \t, "", ".

Literales: true, false

Lista de No Terminales

Definiciones:
<saltoLinea>, <espacio>, <espaciosEnBlanco>, <flotante>, <entero>, <enteroPositivo>, <booleano>, <char>, <string>.
<operadorAritmetico>, <operadorLogico>, <operadorRelacional>, <cAsignacion>, <id>, <delimitador>.
<tipoVar>, <tipoInput>, <tipoFuncion>, <tipoLista>, <tipoListaBase>, <tipoListaTamaño>.
<var>, <varCreacion>, <varAsignacion>, <varCyAsignacion>.
<expresion>, <expresionNumerica>, <expresionChar>, <expresionString>, <expresionBooleana>, <expresionLogica>, <expresionRelacional>.
<expresionAritmetica>, <expresionSuma>, <expresionProducto>, <expresionPotencia>, <expresionFactor>, <expresionAritmeticaUnaria>, <expresionAritmeticaUnariaNegativa>.
<funciones>, <funcion>, <params>, <param>, <bloque>, <sentencias>, <sentencia>.
<estructruraControl>, <decide>, <condicion>, <loop>, <for>, <funcionLlamada>, <return>, <break>, <output>, <output_inline>, <input>.
<varGlobal>, <varGlobalAsignacion>, <varGlobalCreacion>.
<lista>, <declaracionLista>, <creacionYAsignacionLista>, <elementos>, <elementoLista>, <modificarElementoLista>, <obtenerElementoLista>.
<principal>, <programa>.
<comentarios>, <comentario>, <comentarioLinea>, <comentarioBloque>.

Símbolo Inicial:

El símbolo inicial de la gramática es <programa>

Producciones:

Programa:

<programa> ::= <comentarios>? (<varGlobal>*)? <comentarios>? <funciones>? <comentarios>? <principal> <comentarios>?

Función main

<principal> ::= "void" "principal" "є" <expresion> "ɔ" <saltoLinea>* <espaciosEnBlanco> <bloque>

Bloque:

<bloque> ::= ¿ <saltoLinea>* <espaciosEnBlanco> <sentencias> <saltoLinea>* \?

Sentencias:

<sentencias> ::= <sentencia> | <sentencias> <sentencia>

Sentencia:

<sentencia> ::= (<var> | <varAsignacion> | <estructruraControl> | <comentarios> | <funcionLlamada> | <return> | <break> | <output> | <modificarElementoLista> | <lista> | <varGlobal>)

Estructuras de Control:

<estructruraControl> ::= <decide> | <loop> | <for>

Funciones:

<funciones> ::= <funcion> | <funciones> <funcion>

Análisis de Resultados

Lecciones aprendidas:

Eduardo:

- Complejidad de definición de normas gramaticales de un lenguaje.
- Si no se tienen claras las estructuras a conseguir, el trabajo puede ser muy arduo y confuso.

Yosimar:

- A veces hay pequeños detalles en los lenguajes de programación que no notamos, cómo poder sumar números con booleanos, verificar tipos compatibles. Y otras cosas pequeñas.

- Los lenguajes de programación tienen una construcción compleja y muy minuciosa desde la definición de su gramática.

Objetivos alcanzados:

Cada punto general del problema (desde el punto “a” hasta el “s”) fue alcanzado, todo dependerá de detalles que quizá no tomamos en cuenta, pero bajo el desarrollo y a falta de una forma de comprobación, se alcanzó cada objetivo correctamente.

Link del repositorio:

https://github.com/Seraf1n0/BNF-Compiladores