Logotipo, nombre de la empresa

Descripción generada automáticamente **Tecnológico de Costa Rica**

CAL

**Escuela de Ingeniería en Computación**

Compiladores e Interpretes  
Grupo n.º 60

**Profesor**

Allan Rodriguez

**Estudiantes**

Estefani Valverde Marín 2021554564

Jordano Escalante 2018161994

Keingell Moodie 2021495925

**Fecha de entrega**

Viernes 28 marzo 2025

Descripción del problema

Se requiere desarrollar una gramática BNF, incluyendo reglas terminales y no terminales, para un lenguaje de programación imperativo con las siguientes características:

**a.** Permitir la creación de funciones que incluyan estructuras de control, bloques de código delimitados por \ y /, y sentencias finalizadas con ?.

**b.** Manejo de tipos de variables: enteros, flotantes, booleanos, caracteres, cadenas (string) y arreglos estáticos bidimensionales.

**c.** Arreglos de enteros, permitiendo su creación y modificación mediante |, así como su uso en expresiones.

**d.** Permitir sentencias para creación de variables, creación y asignación de expresiones y asignación de expresiones a variables, y algunos casos, sólo expresiones sin asignación.

**e.** Las expresiones pueden combinar literales, variables y funciones de los tipos definidos en la gramática.

**f.** Debe permitir operadores y operandos, respetando precedencia (usual matemática) y permitiendo el uso de paréntesis.

**g.** Permitir expresiones aritméticas binarias de suma (+), resta (-), división (//) –entera o decimal según el tipo--, multiplicación(\*), módulo (~) y potencia (\*\*).Para enteros o reales.

**h.** Permitir expresiones aritméticas unarias de negativo (-), ++, -- , antes del operando; el negativo se puede aplicar a literales enteros y flotantes, el ++ y -- se aplica a variables enteros y flotantes.

i. Permitir expresiones relacionales (sobre enteros y flotantes) de menor, menor o

igual, mayor, mayor o igual, igual y diferente. Los operadores igual y diferente

permiten adicionalmente tipo booleano.

j. Permitir expresiones lógicas de conjunción (^), disyunción (#) y negación (ésta debe

ser de tipo caracter (!)).

k. Debe permitir sentencias de código para las diferentes expresiones mencionadas

anteriormente y su combinación, el delimitador de final de expresión será el

carácter pregunta (?). Además, dichas expresiones pueden usarse en las

condicionales y bloques de las siguientes estructuras de control.

l. Debe permitir el uso de tipos y la combinación de expresiones aritméticas (binarias

y unarias), relacionales y lógicas, según las reglas gramaticales, aritméticas,

relacionales y lógicas del Paradigma Imperativo.

m. La gramática genera un lenguaje con tipado explícito y fuerte.

n. Debe permitir las estructuras de control if-[elif]-[else], do-while y for, además,

permitir return y break. Las expresiones de las condiciones deberán ser valores

booleanos combinando expresiones aritméticas, lógicas y relacionales.

o. Debe permitir las funciones de leer (enteros y flotantes) y escribir en la salida

estándar (cadena carácter, enteros, boolean y flotantes), se pueden escribir literales

o variables.

p. Debe permitir la creación y utilización de funciones, estos deben retornar valores

(entero, flotantes, char y booleanos) y recibir parámetros (con tipo).

q. Debe definir un único procedimiento inicial main, por medio de la cual se inicia la

ejecución de los programas, este es de tipo void.

r. Además, debe permitir comentarios de una línea (@) o múltiples líneas ({ }).

**Diseño del programa**

Se adjunta archivo adicional en la carpeta con el nombre de “gramaticaBNF.txt”

**Análisis de los resultados**

Lecciones aprendidas:

Durante el desarrollo de la gramática BNF para este lenguaje imperativo, se aprendió a identificar y estructurar adecuadamente las reglas terminales y no terminales, permitiendo representar sintácticamente construcciones complejas como funciones, estructuras de control, expresiones aritméticas, lógicas y relacionales. Este proceso permitió profundizar en la lógica gramatical del paradigma imperativo y la precisión que requiere su representación formal.

Objetivos alcanzados

Se lograron todos los objetivos

**Bitácora**

La bitácora puede ser consultada en el repositorio del GitHub utilizando el siguiente enlace:

<https://github.com/Estefani05/Tarea1_Compiladores/tree/main>