Instituto Tecnológico de Costa Rica
Ingeniería en Computación
Compiladores e Intérpretes
Semestre II, 2025
Profesor: Allan Rodríguez Dávila

TAREA #1

Gramática BNF

Lenguaje de programación Imperativo

Introducción

Un grupo de desarrolladores desea crear un nuevo lenguaje imperativo, ligero, que le permita realizar operaciones básicas para la configuración de chips, ya que esta es una industria que sigue creciendo constantemente, y cada vez estos chips necesitan ser configurados por lenguajes más ligeros y potentes. Es por esto que este grupo de desarrolladores requiere desarrollar su propio lenguaje para el desarrollo de sistemas empotrados, y como primer paso necesitan desarrollar una gramática simple y poderosa.

Requerimientos a desarrollar

Se requiere desarrollar una Gramática BNF para un lenguaje imperativo con las siguientes características:

a. Permitir la creación de funciones, y dentro de ellas, estructuras de control, bloques de código (¿ y ?) y sentencias de código.
b. Se permite crear variables globales.
c. Manejar los tipos de variables enteras, flotantes, booleanas, caracteres, cadenas de caracteres (string) y arreglo estático unidimensional.
d. Se permite crear arreglos de tipo char y entero. Además, se permite obtener y modificar sus elementos, y ser utilizados en expresiones. Se permite creación con asignación (¿ y ?).
e. Permitir sentencias para creación de variables (locales y globales), creación y asignación de expresiones y asignación de expresiones a variables, y algunos casos, sólo expresiones sin asignación. En el caso de creaciones utilizar la palabra reservada `let`.
f. Las expresiones permiten combinar literales, variables, arreglos y/o funciones, de los tipos reconocidos en la gramática.
g. Debe permitir operadores y operandos, respetando precedencia (usual matemática) y permitiendo el uso de paréntesis (є y ə).
h. Permitir expresiones aritméticas binarias de suma (+), resta (-), división (// y /) –entera o decimal según el tipo--, multiplicación (*), módulo (%) y potencia (^). Para enteros o reales.

i. Permitir expresiones aritméticas unarias de negativo (-), ++ y -- después del operando; el negativo se puede aplicar a literales enteros y flotantes, el ++ y -- se aplica a variables enteres y flotantes.
j. 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.
k. Permitir expresiones lógicas de conjunción (@), disyunción (~) y negación (ésta debe ser de tipo caracter (Σ)).
l. Debe permitir sentencias de código para las diferentes expresiones mencionadas anteriormente y su combinación, el delimitador de final de expresión será el carácter dólar ($). Además, dichas expresiones pueden usarse en las condicionales y bloques de las siguientes estructuras de control.
m. Debe permitir el uso de tipos y la combinación de expresiones aritméticas (binarias y unarias), relacionales y lógicas, según las reglas gramaticales, aritméticas, relacionales y lógicas del Paradigma Imperativo, por ejemplo, tomando como referencia el lenguaje C. Cualquier duda con respecto al comportamiento de la funcionalidad debe validarlo con el profesor.
n. La gramática genera un lenguaje con tipado explícito y fuerte.
o. Debe permitir las estructuras de control:
    a. **decide of**, de la forma:
    ```
    decide of
    (condicion -> bloque)*
    (else -> bloque)?
    end decide$
    ```
    b. **loop** (tipo Ada, no usa llaves de bloque)
    ```
    loop
    instrucciones…
    exit when condicion$
    end loop$
    ```
    c. **for** (variación de Pascal con to y downto, utiliza llaves después del do para definir un bloque)
    ```
    for asignacion step valor (to o downto) expresion do
    bloque
    ```
    d. Además, permitir **return** y **break**.
    Las expresiones de las condiciones deberán ser valores booleanos combinando expresiones aritméticas, lógicas y relacionales.
p. 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. Utilizar palabras reservadas **output** e **input**.
q. Debe permitir la creación y utilización de funciones, estos deben retornar valores (entero, flotantes, char y booleanos) y recibir parámetros (con tipo).
r. Debe definir un único procedimiento inicial main (llamado `principal`), por medio de la cual se inicia la ejecución de los programas, este es de tipo `void` y no recibe parámetros.

s. Además, debe permitir comentarios de una línea (|) o múltiples líneas (¡ y !).

Aspectos técnicos

1. La tarea es en parejas.

2. La gramática debe ser desarrollada utilizando la notación BNF.

3. Toda gramática libre de contexto BNF debe tener un símbolo inicial, que a partir de ella se generan todas las producciones.

Documentación

La documentación externa deberá incluir:

a. Portada.
b. Descripción del problema.
c. Diseño del programa: Lista de terminales, lista de no terminales, símbolo inicial y producciones.
d. Análisis de resultados: lecciones aprendidas, objetivos alcanzados, objetivos no logrados.
e. Bitácora (blog con control de usuario, hora y fecha): adjuntar evidencia en las entradas. Utilizar git.

Evaluación

La tarea tiene un valor de **5%** de la nota final, en el rubro de Tareas.
Desglose de la evaluación de la tarea:

1. Documentación externa 5 ptos.
2. Funcionalidad 90 ptos (ver detalle en la sección de requerimientos a desarrollar)
3. Hora de Entrega 5 ptos.

Aspectos administrativos

Debe crear un archivo **.zip** (“TC1_Estudiante1_Estudiante2.zip”) que contenga únicamente un archivo **info.txt** y 1 carpeta llamada **documentacion**, que deberá incluir el documento de ***Word*** **o txt** solicitado en la documentación y un archivo con las producciones de la gramática. El archivo **info.txt** debe contener la siguiente información (cualidades):

a. Nombre del curso
b. Número de semestre y año lectivo
c. Nombre de los Estudiantes
d. Número de carnet de los estudiantes
e. Número de tarea

f. Fecha de entrega
g. Estatus de la entrega (debe ser **CONGRUENTE** con la solución entregada): [Deplorable|Regular|Buena|MuyBuena|Excelente|Superior]

Entrega

Deberá subir el archivo antes mencionado al TEC Digital en el curso de COMPILADORES E INTÉRPRETES GR 60, en la asignación llamada "T1" debajo del rubro de "Tareas". En la evaluación de la Tarea el rubro de "Hora de Entrega" valdrá por 5 puntos de la nota total de la tarea, según la siguiente escala:

a. Si se entrega antes de las 11:55:55 **PM** del viernes 12 de septiembre de 2025, 5 puntos.
b. Si se entrega antes de las 11:55:55 **PM** del sábado 13 de septiembre de 2025, 0 puntos.
   Después de este punto, **NO SE ACEPTARÁN** más trabajos.

El archivo será revisado en el sistema de Control de Plagio del TEC Digital. **Todo el contenido debe ser 100% original y en caso de detectar plagio se asignará nota cero a la tarea para todos los estudiantes del grupo y lo indicado en el artículo 75 del RREA.**