Imagen de la pantalla de un celular con letras

Descripción generada automáticamente con confianza bajaForma

**Instituto Tecnológico de Costa Rica**

**Compiladores e Intérpretes**

**Tarea 1: Gramática BNF**

**Profesor:**

**Allan Rodríguez Dávila**

**Estudiantes:**

**Jimena Méndez Morales - 2023113347**

**Ricardo Arce Aguilar - 2023215990**

**I Semestre, 2025**

## Descripción del problema

El objetivo de esta tarea es diseñar una gramática en notación BNF que defina un lenguaje de programación imperativo ligero. Este lenguaje debe soportar variables globales y locales, funciones, expresiones aritméticas, relacionales y lógicas, así como estructuras de control. El diseño busca que el lenguaje sea adecuado para la configuración de sistemas embebidos, manteniendo sintaxis clara y tipado fuerte.

## Diseño del programa

### Lista de terminales

|  |  |
| --- | --- |
| **Categoría** | **Símbolo o palabra reservada** |
| Palabras reservadas | |  | | --- | |  |  |  | | --- | | let, main, decide of, else, end decide, loop, exit when, end loop, for, step, to, downto, do, return, break, input, output | |
| Operadores aritméticos | +, -, \*, /, //, %, ^, ++, -- |
| Operadores relacionales | <, <=, >, >=, ==, != |
| Operadores lógicos | @ (AND), ~ (OR), Σ (NOT) |
| Delimitadores y símbolos | Є, э , $, [, ], ,, =, $ |
| Literales | |  | | --- | |  |  |  | | --- | | Números enteros (0,1,2…), flotantes (3.14), booleanos (true, false), caracteres ('a'), cadenas ("texto") | |
| Comentarios | | , ¡ , ! |

### Lista de no terminales

|  |  |
| --- | --- |
| **No terminal** | **Descripción** |
| programa | Unidad completa del programa. Puede contener declaraciones globales, la función principal y funciones auxiliares. |
| global\_decls | Conjunto de declaraciones globales. Puede ser una secuencia de variables o vacío. |
| global\_decl | Declaración global de una variable con su tipo, identificador, opcionalmente arreglo e inicialización. |
| funciones | Conjunto de funciones definidas por el usuario. Puede ser una sola o varias en secuencia. |
| funcion | Definición de una función con tipo de retorno, nombre, parámetros y bloque de instrucciones. |
| principal | Definición del punto de entrada del programa (principal) con un bloque de instrucciones. |
| parametros | Lista de parámetros de una función. Puede ser vacía o una secuencia separada por comas. |
| parametro | Un parámetro individual: tipo e identificador, opcionalmente un arreglo. |
| tipo | Tipos de datos básicos admitidos (int, float, char, bool, string). |
| opt\_array\_decl | Especifica si una **variable** es un arreglo ([n]) o no. |
| opt\_array\_param | Especifica si un parámetro de función es un arreglo ([n]) o no. |
| opt\_init | Inicialización opcional de una variable (= expr) o vacío. |
| declaracion | Declaración de variables locales dentro de un bloque, con tipo, id, y opcionalmente arreglo e inicialización. |
| asignacion | Sentencia de asignación, ya sea simple, acceso a un elemento de arreglo o llamada a función. |
| llamada\_func | Invocación de una función con argumentos. |
| args | Lista de argumentos en una llamada a función, puede estar vacía. |
| control | Conjunto de estructuras de control de flujo: decideof, loop, for. |
| decideof | Estructura condicional múltiple (similar a switch o case). |
| loop | Estructura de repetición loop ... exit when ... end loop. |
| for | Ciclo for con inicialización, paso, dirección (to/downto) y bloque. |
| expr | Expresiones, con precedencia definida (lógica, relacional, aritmética, unarios, primarios). |
| instruccion | Una instrucción individual: declaración, asignación o estructura de control. |
| bloque | Secuencia de instrucciones, que puede ser vacía. |
| numero | Constante entera. |
| floatnum | Constante numérica con decimales. |
| booleano | Literal booleano (true o false). |
| cadena | Literal de cadena delimitada por comillas dobles. |
| caracter | Literal de carácter delimitado por comillas simples. |
| id | Identificador para variables, funciones o parámetros. |

### Símbolo inicial

programa

### Producciones

programa ::= global\_decls principal funciones

programa ::= global\_decls funciones principal

programa ::= global\_decls funciones principal funciones

global\_decls ::= global\_decl global\_decls | ε

global\_decl ::= "let" tipo id opt\_array\_decl opt\_init $

funciones ::=funcion

funciones ::= funcion funciones

funcion ::= tipo id є parametros э ¿ bloque ?

principal ::= "principal" є э ¿ bloque ?

parametros ::= parametro ("," parametro)\* | ε

parametro ::= tipo id

parametro ::= tipo id opt\_array\_param **Parámetros arreglos como (int arr[10])**

tipo ::= "int" | "float" | "char" | "bool" | "string"

opt\_array\_decl ::= "[" numero "]" | ε

opt\_array\_param ::= "[" numero "]" | ε

opt\_init ::= "=" expr | ε

declaracion ::= "let" tipo id opt\_array\_decl opt\_init

declaracion ::= "let" tipo id opt\_array\_decl

asignacion ::= id "=" expr

asignacion ::= id "[" expr "]" "=" expr

asignacion ::= llamada\_func

llamada\_func ::= id є args э

args ::= expr (“,” expr)\* | ε

control ::= decideof | loop | for

decideof ::= "decide of" ( є expr э "->" ¿ bloque ? )\* ( "else" "->" ¿ bloque ? )? "end decide" $

loop ::= "loop" ¿ bloque ? "exit when" expr $ "end loop" $

for ::= "for" asignacion "step" numero ("to" | "downto") expr "do" ¿ bloque ?

expr ::= logexpr **Expresiones con precedencia (lógica -> relacional -> aritmética -> unarios -> primarios)**

instruccion ::= declaracion $ | asignacion $ | control **Principal para usarla en el bloque**

bloque ::= instruccion bloque | ε

numero ::= [0-9]+ -- enteros

floatnum ::= [0-9]+"."[0-9]+ -- números con decimales

booleano ::= "true" | "false"

cadena ::= "\"" texto "\"" -- entre comillas dobles

caracter ::= "'" . "'" -- un solo carácter

id ::= [a-zA-Z\_] ([a-zA-Z\_] | [0-9])\*

## Análisis de resultados

### Lecciones aprendidas

* Se logró cubrir las principales características de un lenguaje imperativo (tipos, expresiones, funciones y control de flujo).
* Como mejora futura, se podría detallar aún más la semántica de operadores y la gestión de arrays.
* El objetivo principal de tener un lenguaje imperativo ligero se cumplió.

### Objetivos

|  |  |  |
| --- | --- | --- |
| **Objetivo** | **Alcanzado** | **No alcanzado** |
| Diseñar una gramática en notación BNF |  |  |
| Permitir la creación de funciones y un procedimiento principal (main) |  |  |
| Incorporar estructuras de control (decide of, loop, for) |  |  |
| Definir bloques de código con ¿ y ? y expresiones agrupadas con є y э |  |  |
| Soportar variables globales y locales |  |  |
| Soportar variables globales y locales |  |  |
| Definir expresiones aritméticas, relacionales y lógicas |  |  |
| Incluir sentencias de entrada y salida |  |  |
| Permitir comentarios de una y varias líneas |  |  |
| Asegurar que la gramática pueda generar programas válidos |  |  |

## Bitácora

<https://github.com/jimendezm/tarea1-ce>