**[[1]](#footnote-1) 1. INTRODUCCIÓN**

Intérprete Interactivo para Enseñar Lógica Algorítmica: Un Enfoque Basado en Lightbot

**Pedroza Palomar, Oriana1; Hernández, Juan1**

1 Escuela de Ingeniería de Sistemas, Facultad de Ingeniería, Universidad de Los Andes, Mérida, Venezuela.

**Resumen:** -

**Palabras clave**: Scanner, lenguaje de programación, compilador.

El objetivo de este proyecto es desarrollar un intérprete para el lenguaje de programación no tipado diseñado para enseñar lógica algorítmica fundamental de manera interactiva y atractiva, inspirado en el juego *Lightbot*. Este lenguaje permite a los usuarios controlar un robot mediante una serie de comandos para realizar tareas específicas y resolver problemas lógicos.

En la implementación de nuestro intérprete, comenzaremos con el desarrollo del scanner, también conocido como analizador léxico. Esta es la primera etapa del proceso de compilación. Durante esta fase, el scanner lee el código fuente y convierte la entrada en tokens, basados en un conjunto de reglas definidas para el lenguaje. Cada token representa un componente sintáctico del lenguaje, como palabras clave, identificadores, números, operadores y delimitadores. Esta etapa es crucial para transformar el código fuente en una forma que puede ser procesada por las etapas siguientes del intérprete.

**LENGUAJE DE PROGRAMACIÓN**

**Languages Semantics:**

**Expressions:**

* Int: This expression is an integer.
* LoopStep: The loopstep expression represents the current step of a loop as an expression.
* Equal: Represents an equality comparison between two expressions.

**Statements:**

* move: this statement receives the environment reference as its argument. The execute expression of the move statement is responsible for moving the character one step forward in its current direction within the environment.
* Turn\_left: this statement operates over the character direction store within the environment. The execute function of the turn left statement is responsible for turning the character 90 degrees counterclockwise.
* Turn right: this statement operates over the character direction store within the environment, receiving an Environment& reference. The execute function of the turn right statement is responsible for turning the character 90 degrees clockwise.
* Loop: This class defines a structure to execute a block of statements a specified amount of times. The attribute times represents how many times the loop should repeat.
* Finish: Represents the ending statement for the program.
* If: Represents a conditional statement in the program.

**SCANNER**

En esta etapa se procede a definir cuáles serán los tokens que se utilizarán en las etapas subsiguientes del proceso de compilación. Por esta razón y según las características del leguaje se necesitan reconocer tokens para movimientos, giros, encendido de luz, bucles, condicionales, procedimientos, entre otros. A continuación, se plantean los siguientes tokens definidos para su reconocimiento en el proceso de análisis léxico:

\* TOKEN\_MOVE \* TOKEN\_TURN\_LEFT

\* TOKEN\_TURN\_RIGHT \*TOKEN\_LIGHT\_UP

Estos tokens representan comandos básicos que permiten al robot realizar acciones específicas como por ejemplo moverse siempre de frente, voltear a la izquierda o derecha y encender la luz para finalizar.

\* TOKEN\_LOOP \* TOKEN\_IF \* TOKEN\_ELSE

Representan estructuras de control, como el uso de los bucles y las decisiones basadas en condiciones.

\* TOKEN\_PROCEDURE \* TOKEN\_CALL

Son necesarios para definir y utilizar procedimientos en el lenguaje.

\* TOKEN\_LEFT\_PAREN \* TOKEN\_RIGHT\_PAREN

\* TOKEN\_LEFT\_BRACE \* TOKEN\_RIGHT\_BRACE

\* TOKEN\_COMMA \* TOKEN\_EQUAL

Estos tokens ayudan a estructurar el código y a separar los elementos de las instrucciones, son conocidos como los delimitadores ‘()’, ‘{}’ y los operadores ‘,’ ‘=’.

También se incluyen los tokens TOKEN\_EOF, TOKEN\_INT, TOKEN\_IDENTIFIER, los cuales son fundamentales para identificar el final del archivo, los enteros y los identificadores del lenguaje.

Validator

El archivo parse.bison consta de:

program:

El programa comienza con un procedimiento definido por la palabra clave PROCEDURE. El resultado del análisis sintáctico se almacena en parser\_result.

procedure:

Define un procedimiento que incluye un nombre (el identificador) y un bloque de comandos delimitado por {}. Este bloque puede contener varias acciones o comandos que el robot debe ejecutar.

commands y command:

Se define una lista de comandos. Los comandos son acciones que el robot puede realizar, como MOVE(), TURN\_LEFT(), TURN\_RIGHT(), LOOP, y CALL.

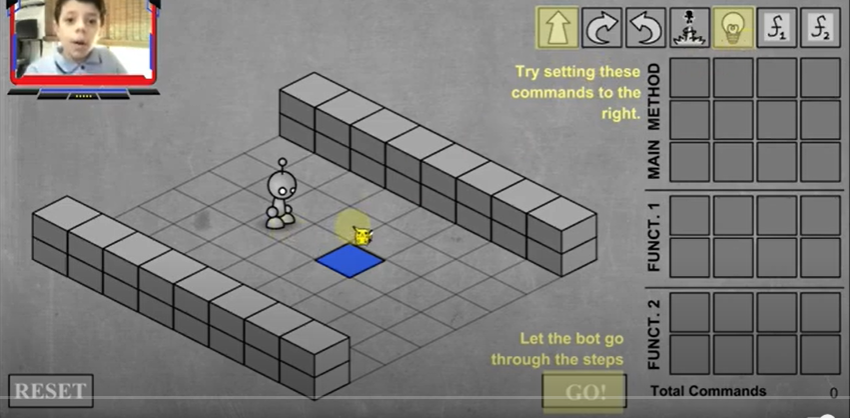
Cada acción se convierte en una instancia de una clase de comando correspondiente, por ejemplo, Move, TurnLeft, Loop, etc.

LOOP: El bucle LOOP toma un número de iteraciones y ejecuta un bloque de comandos varias veces.

CALL: Invocar procedimientos definidos por el usuario con la palabra clave CALL.

**CONCLUSIONES**

-

****

**REFERENCIAS**

Lightbot. (n.d.). Lightbot: Learn to code by solving puzzles. Retrieved July 18, 2024, from <https://lightbot.com/>

Mujica, Alejandro. (2024). Lecture01. Obtenido de: <https://docs.google.com/presentation/d/1OkOJO4Ppw0JdPZ6m6bZC6EDXWgsUrmnqe5ai-GWzb4A/edit?usp=sharing>. (Julio, 2024).

Mujica, Alejandro. (2024). Lecture02. Obtenido de: <https://docs.google.com/presentation/d/1OkOJO4Ppw0JdPZ6m6bZC6EDXWgsUrmnqe5ai-GWzb4A/edit?usp=sharing>. (Julio, 2024).

Mujica, Alejandro. (2024). Lecture03. Obtenido de: <https://docs.google.com/presentation/d/1OkOJO4Ppw0JdPZ6m6bZC6EDXWgsUrmnqe5ai-GWzb4A/edit?usp=sharing>. (Julio, 2024).

Rasmos, R., Salas, J., & Hernández, J. (2024). Programming languages: Programmer’s playground project. Retrieved July 18, 2024, from <https://docs.google.com/document/d/1MKd55Z-GijVhqKZsJWUnXSrMCQTPV6o8EJCxd81vUcs/edit#heading=h.5qhyiwmbjyq2>

1. [pedrozapalomar.ov07@gmail.com](mailto:pedrozapalomar.ov07@gmail.com) / @gmail.com [↑](#footnote-ref-1)