Universidad de Los Andes, Departamento de Ingeniería de Sistemas
ISIS 1106- Lenguajes y Máquinas.
18 de febrero de 2024
P0 – Entrega 1
Grupo 11
Juan David Duarte 2022155070
Natalia Isabela Erazo González 202213680

P0 Lenguajes y máquinas

Juan David Duarte – 202215070

Natalia Isabela Erazo - 202213680

Enlace github: https://github.com/LYM-2024-01/P0-LYM.git

El analizador del lenguaje del robot se separó en tres documentos python.

main.py: Acá se pregunta por la dirección de memoria del archivo que se desea leer, desde acá se llama a los archivos lexer.py y parse.py.

lexer.py: En este archivo se separan las palabras del archivo en una lista, pero ahora se categoriza cada palabra con ayuda de algunas palabras reservadas que pertenecen a alguna categoría si son constantes, como los siguientes casos: ((Los nombres de los TOKEN's son los mismos de las listas que los contienen)

```
NUMBERS = ["1", "2", "3", "4", "5", "6", "7", "8", "9", "0"]
CONSTANTES = ["dim", "myxpos", "myypos", "mychips", "myballoons", "balloonshere", "chipsHere", "spaces"]
O = [":north", ":south", ":east", ":west"]
D = [":left", ":right", ":around", ":front", ":back", ":up", ":down"]
X = [":balloons", ":chips"]
NULL = ["null"]
```

Luego se clasifican los comandos, las estructuras y las condiciones que cuyo toquen tiene el mismo nombre que ellas, pero en mayúsculas. Estas son:

```
SUBCOMANDOS = ["defvar", "move", "skip", "turn", "face", "put", "pick", "move-dir", "run-dirs", "move-face", "null"]

SUBESTRUCTURA = ['if', 'loop', 'repeat', 'defun']

SUBCONDICIONES = ['facing?', 'blocked?','can-put?', 'can-pick?', 'can-move?', 'iszero?', 'not']
```

parser.py: En este archivo python se evalúa si el texto una vez tokenizado es válido según las reglas del lenguaje.

Acá se itera sobre la lista de tokens y se van guardando uno a uno en una pila y se guarda la posición de los paréntesis que abren "(" en la pila y se itera hasta que encuentre paréntesis que cierran ")" y con esto se analiza el bloque de paréntesis desde el ultimo paréntesis que abre

hasta el que cierra y revisa si ese bloque es una instrucción de la gramática o de lo contrario sale del programa e imprime que la gramática no está bien. Si la instrucción está bien se le asigna un token dependiendo de su tipo pertenece, si es un comando se le asigna el token “COMANDO”, si es una condición se deja como “CONDICION”, etc...

Si hay algún error, es porque hay algo del lenguaje que está mal, se lanza una excepción y se imprime que el lenguaje está mal.

Si todo esta no hay ningún error y si todas las expresiones están bien, se imprime que el lenguaje está bien.