

**INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY**

**Implementación de métodos computacionales (Gpo 820)**

**Prof. Román Martínez Martínez**

**José Carlos Zertuche de la Cruz**

**A01198177**

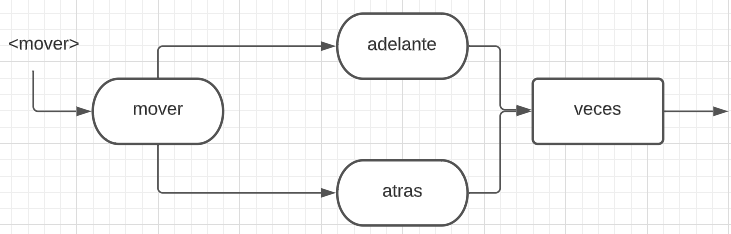
**20 de marzo del 2022**

**Evidencia #1: Diseño e implementación básica de un DSL para enseñar a programar a niños**

**Elementos del léxico del lenguaje y diagramas de sintaxis**

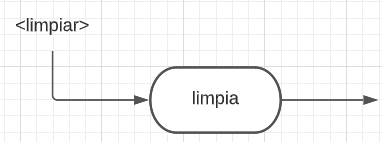
**Mover**

* **Expresión regular**: m\_(adelante | atras)\_(\[0-50]\)
* **Lexemas**: “m”, “adelante”, “atras”
* **Ejemplo**: m\_atras\_[20]
  + Quiere decir que se mueva 20 veces (o pasos) para atrás.



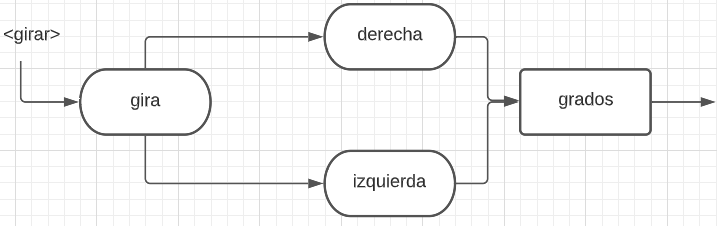
**Limpiar**

* **Expresión regular**: limpia
* **Lexemas**: “limpia”
* **Ejemplo**: limpia
  + Borra toda la pantalla



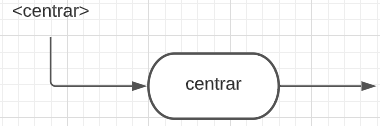
**Girar**

* **Lexemas**: “gira”, “der”, “izq”
* **Expresión regular**: gira\_(izq | der)\_(\[0-360]\)
* **Ejemplo**: gira\_izq\_[180]
  + Quiere decir que el objeto gire 180 grados a la izquierda, NO se mueve, solo gira su orientación.



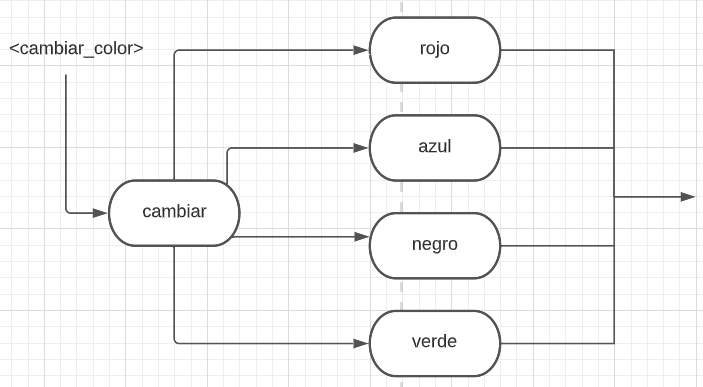
**Centrar**

* **Lexemas**: “centrar”
* **Expresión regular**: centrar
* **Ejemplo**: centrar
  + Pondrá al robot en medio de la pantalla



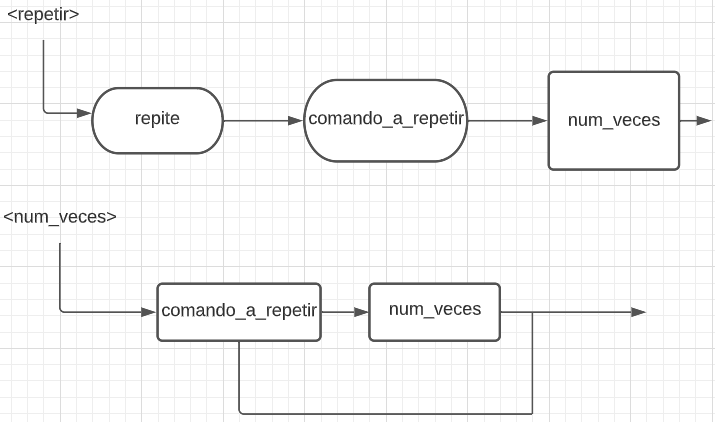
**Cambiar color pluma**

* **Lexemas**: “cambiar”, “rj”, “az”, “ver”, “ng”
* **Expresión regular**: cambiar\_(rj | az | ver | ng)
* **Ejemplo**: cambiar\_ver
  + Cambiará el color de la pluma a verde



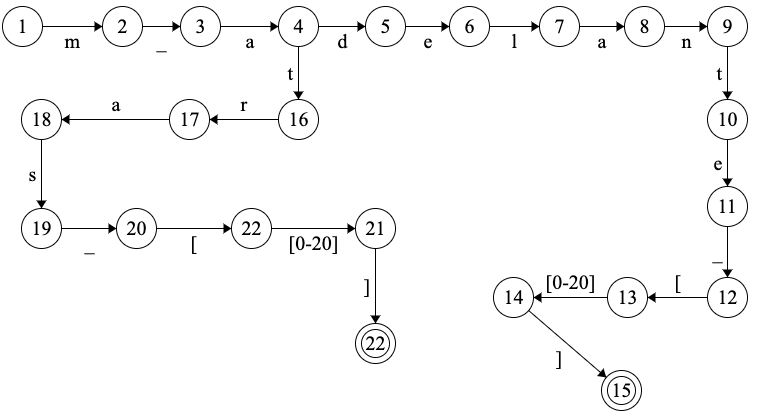
**Repetir comando**

* **Lexemas**: “repite”, “*comando\_previamente\_conocido*”
* **Expresión regular**: repite\_<num\_veces>\_(comando\_a\_repetir)
* **Ejemplo**: repite\_4\_m\_adelante\_[10]
  + Repetirá 15 veces el comando mover adelante 4 unidades.

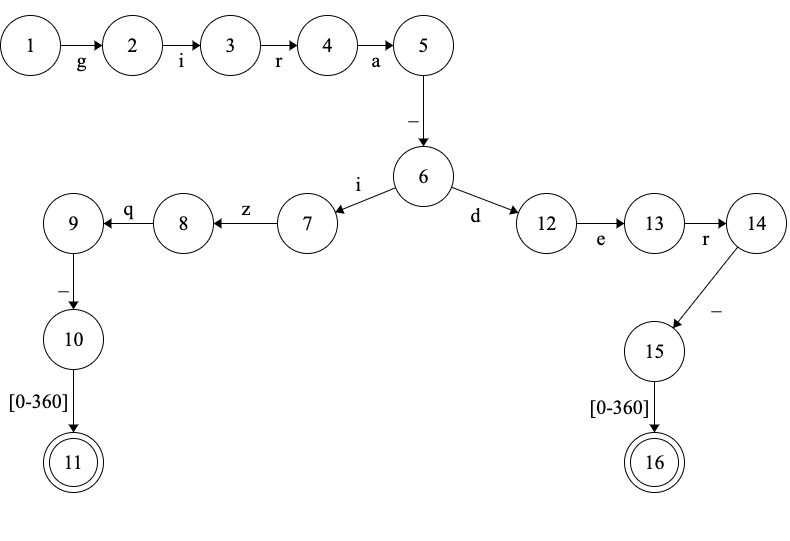


**Autómata determinístico**

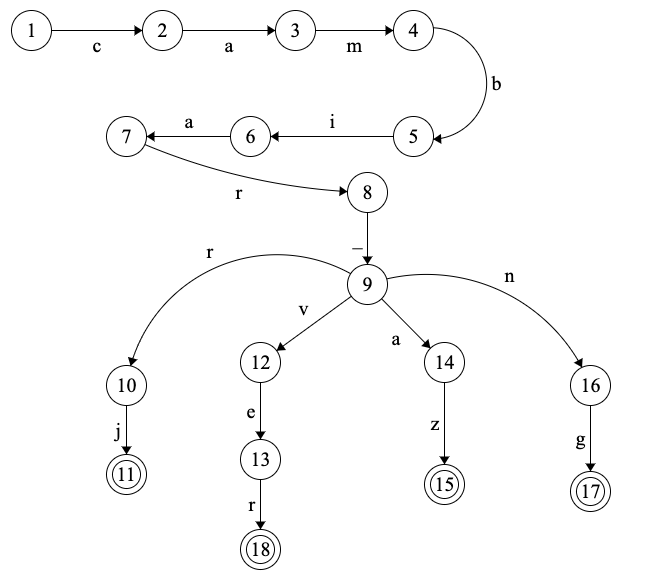
**MOVER**

****

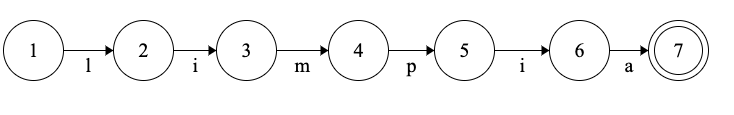
**GIRAR**

****

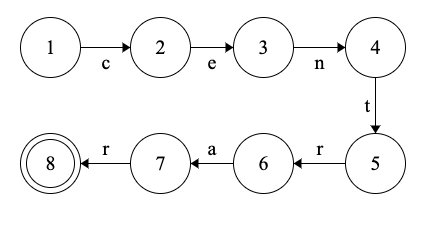
**CAMBIAR COLOR DE PLUMA**

****

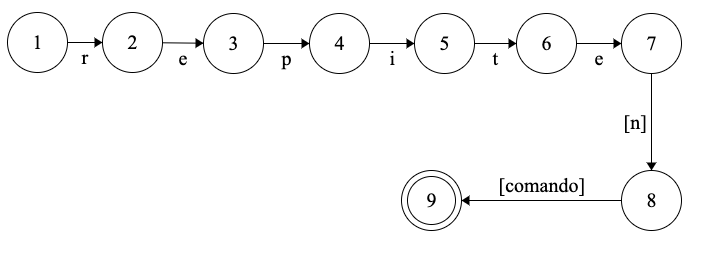
**LIMPIAR LA PANTALLA**

****

**CENTRAR**

****

**REPETIR COMANDO**

****

**Gramática BNF**

**(Cada guión bajo, representa un espacio en blanco)**

<comando> ::= <mover> | <limpiar> | <girar> | <centrar> | <cambiar\_color> | <repetir>

<mover> ::= "m"\_<direccion>\_<distancia>

<direccion> ::= "adelante" | "atras"

<distancia> ::= "["\_<numero>\_"]"

<numero> ::= <digito><numero> | <digito>

<digito> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

<limpiar> ::= "limpia"

<girar> ::= "gira"\_<direccion\_giro>\_<angulo>

<direccion\_giro> ::= "der" | "izq"

<angulo> ::= "["\_<grados>\_"]"

<grados> ::= <digito><grados> | <digito>

<centrar> ::= "centrar"

<cambiar\_color> ::= "cambiar"\_<color>

<color> ::= "rj" | "az" | "ver" | "ng"

<repetir> ::= "repite"\_<num\_veces>\_<comando\_a\_repetir>

<num\_veces> ::= <digito><num\_veces> | <digito>

<comando\_a\_repetir> ::= <mover> | <limpiar> | <girar> | <centrar> | <cambiar\_color>

**Codigo implementado**

**#Autor: Jose Carlos Zertuche de la Cruz**

**#Evidencia 1**

**#Implementando un minilenguaje de programacion**

**#Nombre del lenguaje: Zerlang**

**import turtle**

**import re**

**def mover(direccion, distancia): #funcion de mover**

**if direccion == "adelante":**

**turtle.forward(distancia)**

**elif direccion == "atras":**

**turtle.backward(distancia)**

**def limpiar(): #funcion de limpiar la pantalla**

**turtle.clear()**

**def girar(direccion, angulo): #funcion de girar el robot**

**if direccion == "izq":**

**turtle.left(angulo)**

**elif direccion == "der":**

**turtle.right(angulo)**

**def centrar(): #funcion de centrar el robot**

**turtle.home()**

**def cambiar\_color(color): #funcion de cambiar color del robot**

**if color == "rj":**

**turtle.pencolor("red")**

**elif color == "az":**

**turtle.pencolor("blue")**

**elif color == "ver":**

**turtle.pencolor("green")**

**elif color == "ng":**

**turtle.pencolor("orange")**

**def repetir(num\_veces, comando\_a\_repetir): #funcion de repetir el comando**

**for i in range(num\_veces):**

**ejecutar\_comando(comando\_a\_repetir)**

**def ejecutar\_comando(token): #funcion principal utilizando el descenso recursivo**

**#Este primer if verifica si el token tiene el formato "m\_(adelante|atras)\_(n)", donde n es la cantidad**

**#de pasos que se movera. Si se cumple el formato, extrae la dirección, la distancia y llama a la funcion mover**

**#para mover al robot**

**pasos\_maximos = 50 #lo maximo que podra avanzar el robot sera 50**

**if re.match(r"m\_(adelante|atras)\_(\[\d+\])", token):**

**direccion, distancia = re.findall(r"m\_(adelante|atras)\_(\[\d+\])", token)[0]**

**if direccion == "adelante":**

**if int(distancia[1:-1]) > pasos\_maximos:**

**print(f"Error: la distancia máxima para mover hacia adelante es {pasos\_maximos}")**

**else:**

**mover(direccion, int(distancia[1:-1]))**

**elif direccion == "atras":**

**if int(distancia[1:-1]) > pasos\_maximos:**

**print(f"Error: la distancia máxima para mover hacia atrás es {pasos\_maximos}")**

**else:**

**mover(direccion, int(distancia[1:-1]))**

**#Este segundo if verifica si el token es igual a "limpiar"**

**elif token == "limpia":**

**limpiar()**

**#Se verifica si tiene formato gramatical de "gira\_(izq|der)\_(n)", donde n es el angulo al cual rotara el robot.**

**elif re.match(r"gira\_(izq|der)\_(\[\d+\])", token):**

**direccion, angulo = re.findall(r"gira\_(izq|der)\_(\[\d+\])", token)[0]**

**girar(direccion, int(angulo[1:-1]))**

**#Verifica si el token es igual a centar, y llama a la funcion centrar**

**elif token == "centrar":**

**centrar()**

**#Se verifica si el token es igual a "cambiar\_(rj|az|ver|ng)", si si cumple se llama a la funciond de cambiar el color.**

**elif re.match(r"cambiar\_(rj|az|ver|ng)", token):**

**color = re.findall(r"cambiar\_(rj|az|ver|ng)", token)[0]**

**cambiar\_color(color)**

**#Se verifica si el token tiene el formato "repite\_(n)\_(comando)", donde n indica la cantidad que se repetira el comando.**

**#Si se cumple el formato, guarda el número de veces, el comando a repetir, y se llama la función repetir.**

**elif re.match(r"repite\_(\d+)\_(.+)", token):**

**num\_veces, comando\_a\_repetir = re.findall(r"repite\_(\d+)\_(.+)", token)[0]**

**repetir(int(num\_veces), comando\_a\_repetir)**

**#La funcion valida y ejecuta comandos dependiendo si identifica uno de los formatos definidos en mi lenguaje,**

**#o caso contrario, muestra error si la sintaxis no es la correcta o si hay algun error de por medio.**

**def analizar\_linea(linea):**

**token\_valido = False**

**for frase in ["m\_(adelante|atras)\_(\[\d+\])",**

**"limpia",**

**"gira\_(izq|der)\_(\[\d+\])",**

**"centrar",**

**"cambiar\_(rj|az|ver|ng)",**

**"repite\_(\d+)\_.+"]:**

**if re.match(frase, linea):**

**token\_valido = True**

**ejecutar\_comando(linea)**

**break**

**if not token\_valido:**

**print(f"Error de sintaxis: '{linea}' no es un comando válido")**

**#Esto lo hice para que no se tenga que correr el programa una y otra vez siempre.**

**while True:**

**linea = input("Ingrese un comando: ")**

**if linea == "salir":**

**break**

**analizar\_linea(linea)**

**turtle.done()**

**Comentario sobre la experiencia de aprendizaje y los resultados obtenidos.**

La verdad que quedé sorprendido con lo relativamente “fácil” que es hacer un lenguaje de programación ya que con las librerías de python es mucho más sencillo hacer la interfaz gráfica. Esa era mi duda. Aprendí bastante sobre el proceso detrás de un lenguaje de programación y los tipos que existen y más o menos como operan desde cero. Es un tema que me gustó mucho aunque claramente tiene su grado de complejidad, pero estoy satisfecho con la efectividad que tuve y que si me pudo funcionar correctamente. Quedo muy agradecido con los profesores que siempre nos apoyaron.

Video a youtube

<https://youtu.be/fw45iAU3A4E>