Instituto Tecnológico de Estudios Superiores Monterrey
Campus Querétaro

[TC2037.820 Implementación de métodos computacionales]

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

Profesor(es):

Román Martínez Martínez
María Valentina Narváez Terán
Alberto Oliart Ros
Germán Domínguez Solís

Presenta:

Ian Joab Padrón Corona A01708940

Repositorio en GitHub del Proyecto:
https://github.com/Ian326/TC2037

Léxico del Lenguaje

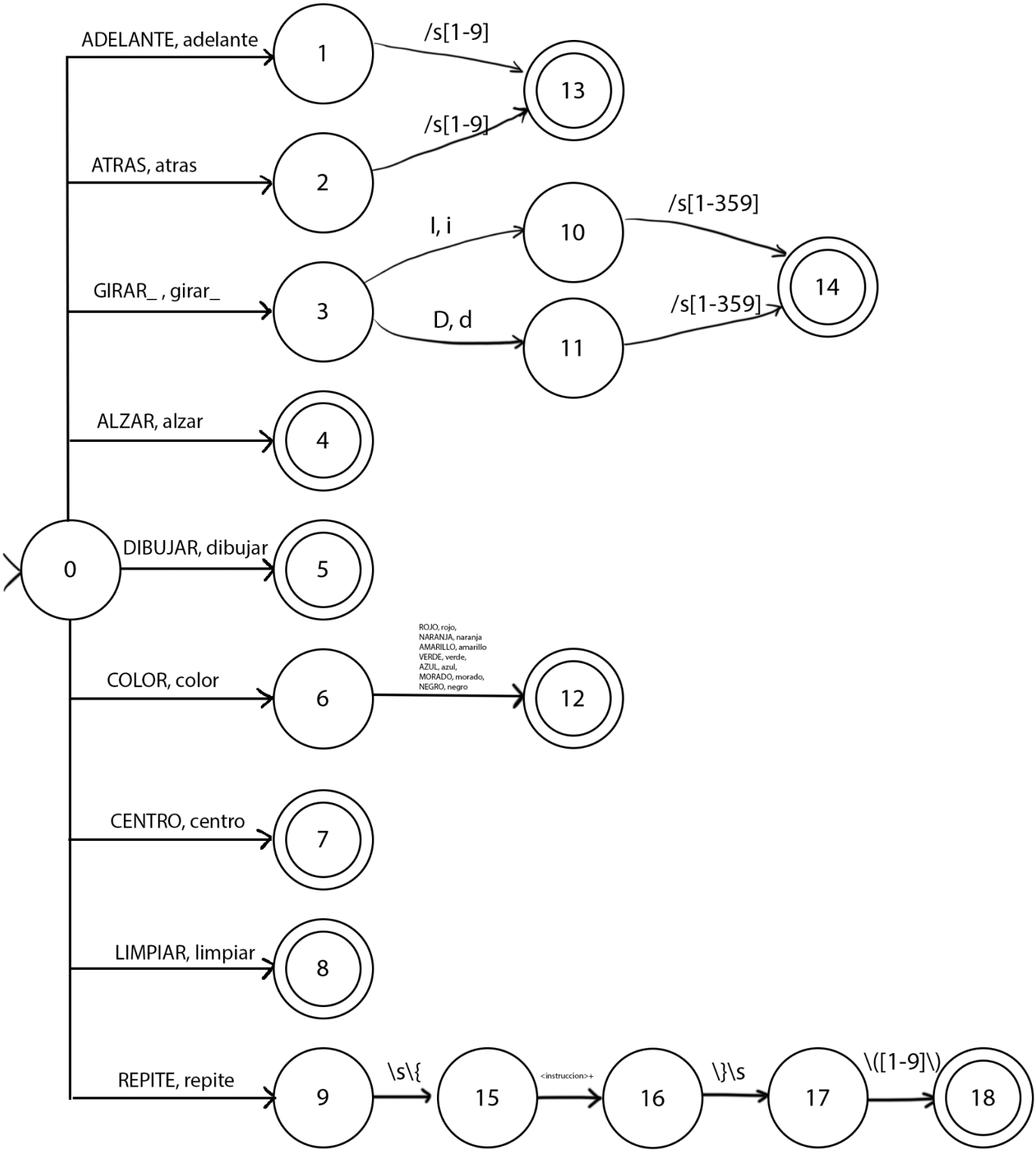
Palabras clave:

- ‘adelante’
- ‘ADELANTE
- ‘atras
- ‘ATRÁS
- ‘gira_i’
- ‘GIRA_I’
- ‘gira_d’
- ‘GIRA_D’
- ‘alzar’
- ‘ALZAR’
- ‘dibujar’
- ‘DIBUJAR’
- ‘color’
- ‘COLOR’
- ‘centro’
- ‘CENTRO’
- ‘limpiar’
- ‘LIMPIAR’
- ‘repite’
- ‘REPITE’

Expresiones regulares para las palabras clave:

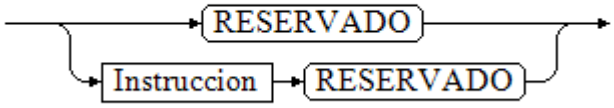
Palabras clave	RegEx
ADELANTE , adelante	(ADELANTE|adelante)
ATRAS , atras	(ATRAS|atras)
GIRA_I , gira_i	(GIRA_I|gira_i)
GIRA_D , gira_d	(GIRA_D|gira_d)
ALZAR , alzar	(ALZAR|alzar)
DIBUJAR , dibujar	(DIBUJAR|dibujar)
COLOR , color	(COLOR|color)
CENTRO , centro	(CENTRO|centro)
LIMPIAR , limpiar	(LIMPIAR|limpiar)
REPITE , repite	(REPITE|repite)

Autómata Determinístico

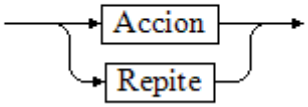


Diagramas de Sintaxis

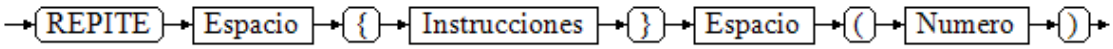
<Instrucciones>



<Instruccion>

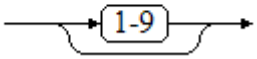


<Repite>

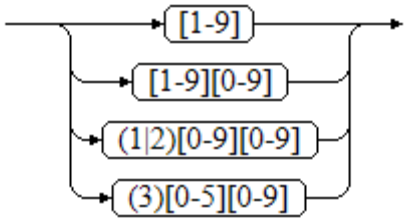


<Espacio>

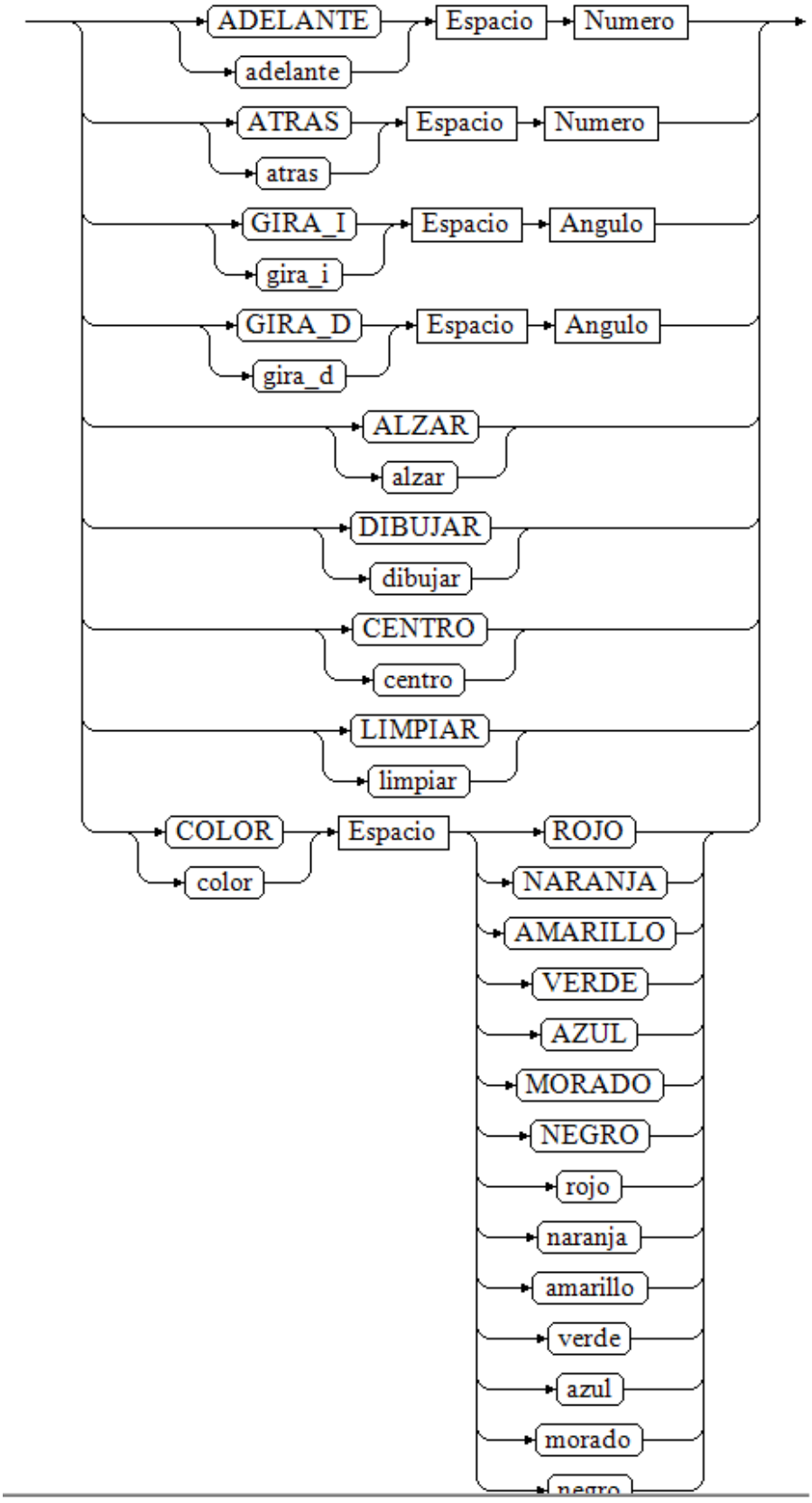
<Numero>



<Angulo>



〈Accion〉


ADELANTE
adelante
Espacio
Numero
ATRAS
atras
Espacio
Numero
GIRA_I
gira_i
Espacio
Angulo
GIRA_D
gira_d
Espacio
Angulo
ALZAR
alzar
DIBUJAR
dibujar
CENTRO
centro
LIMPIAR
limpiar
COLOR
color
Espacio
ROJO
NARANJA
AMARILLO
VERDE
AZUL
MORADO
NEGRO
rojo
naranja
amarillo
verde
azul
morado
negro

Gramáticas BNF

<Instrucciones> ::= (<RESERVADO> | <Instrucción> RESERVADO).

<Instrucción> ::= (<Acción> | <Repite>).

<Acción> ::= ((("ADELANTE"|"adelante") <Espacio> <Numero>)

| (("ATRAS"|"atras") <Espacio> <Numero>))

| (("GIRA_I"|"gira_i") <Espacio> <Angulo>)

| (("GIRA_D"|"gira_d") <Espacio> <Angulo>)

| ("ALZAR"|"alzar")

| ("DIBUJAR"|"dibujar")

| ("CENTRO"|"centro")

| ("LIMPIAR"|"limpiar")

| (("COLOR"|"color") <Espacio> ("ROJO" | "NARANJA" | "AMARILLO" | "VERDE" | "AZUL" | "MORADO" | "NEGRO" | "rojo" | "naranja" | "amarillo" | "verde" | "azul" | "morado" | "negro"))).

<Repite> ::= "REPITE" <Espacio> "{"<Instrucciones>"}" <Espacio> "("<Numero>")".

<Espacio> ::= " ".

<Angulo> ::= ("[1-9]" | "[1-9][0-9]" | "(1|2)[0-9][0-9]" | "(3)[0-5][0-9]").

<Numero> ::= ["1-9"].

Código Implementado

```
'''
    // =========================================================
    // File: miniLOGO
    // Author: Ian Joab Padrón Corona - A01708940
    // Date: 20/03/2023
    // =========================================================
'''
# Este programa ocupa la librería 'Py-cairo'. Pasos para instalación se muestran a continuación:
# 1. Instalar librería base
#   pip3 install pycairo

# 2. (Solo para Ubuntu/Debian) Instalar 'pkg-config' y 'cairo' con headers
#   sudo apt install libcairo2-dev pkg-config python3-dev

#Librerías importadas
import re
import os
import math

import cairo

#Aqui se inicializa el objeto o Canva de 'Py-cairo'
FILE_NAME = os.path.basename(__file__)
WIDTH = 2000                                                    #Dimensiones del Canva
HEIGHT = 2000                                                   #Dimensiones del Canva
ims = cairo.ImageSurface(cairo.FORMAT_ARGB32, WIDTH, HEIGHT)
cr = cairo.Context(ims)

#Aqui se dibuja el Canva Inicial (Un rectángulo rellenado con color blanco)
cr.set_source_rgba(1,1,1,1)
cr.rectangle(
    0,0,
    WIDTH,HEIGHT
)
cr.fill()

#Aquí se declara el color (negro) y grosor (10) default de la pluma
cr.set_source_rgba(0,0,0,1)
cr.set_line_width(10)

#Variables globales de la pluma
angulo = 90*(math.pi/180)
radio = 100
color = [0,0,0,1]
pos = [WIDTH/2,HEIGHT/2]

#Aquí se define la posición inicial de dibujado (El Centro del Canvas <0,0>)
cr.move_to(
    pos[0],
    pos[1]
)

#Aquí se definen las entradas válidas como un diccionario de expresiones regulares
BNF = {
    "adelante": r"(ADELANTE|adelante)\s[1-9]",
    "atras": r"(ATRAS|atras)\s[1-9]",
    "girarIzquierda": r"(GIRA_I|gira_i)\s(((1|2|3)[0-5][0-9])|[1-9][0-9]|[1-9])",
    "girarDerecha": r"(GIRA_D|gira_d)\s(((1|2|3)[0-5][0-9])|[1-9][0-9]|[1-9])",
    "alzarPluma": r"(ALZAR|alzar)",
    "bajarPluma": r"(DIBUJAR|dibujar)",
    "cambiarColor": r"(COLOR|color)\s(ROJO|rojo|NARANJA|naranja|AMARILLO|amarillo|VERDE|verde|AZUL|azul|MORADO|morado|NEGRO|negro)",
    "limpiarPantalla": r"(LIMPIAR|limpiar)",
    "centrarPluma": r"(CENTRO|centro)",
    "repite":   r"(REPITE|repite)\s\{(((ADELANTE|adelante)\s[1-9]|(ATRAS|atras)\s[1-9]|((GIRA_I|gira_i)\s(((1|2|3)[0-5][0-9])|[1-9][0-
9]|[1-9]))|((GIRA_D|gira_d)\s(((1|2|3)[0-5][0-9])|[1-9][0-9]|[1-
9]))|(ALZAR|alzar)|(DIBUJAR|dibujar)|((COLOR|color)\s(ROJO|rojo|NARANJA|naranja|AMARILLO|amarillo|VERDE|verde|AZUL|azul|MORADO|morado|
NEGRO|negro))|(CENTRO|centro))(\s)?)+\}\s\([1-9]\)"
}

# Variable global del repetir()
iteracion = 0
iteracionmax = 9


# Función para validar una cadena de entrada
def validar_entrada():

    print("Leyendo instrucciones...")

    with open("instrucciones.txt") as archivo:
        ln_num = 1
        for input_str in archivo:
            find_match = False
            for key, value in BNF.items():
                # Buscar una coincidencia de expresión regular
                match = re.match(value, input_str)

                # Si hay una coincidencia, la cadena es válida y llama a la función 'instruccion()'
                if match:
                    print(f"La expresión en la línea {ln_num} es válida. Gramática reconocida: '{key}'\n")
                    find_match = True
```

```
                instArray = input_str.split()
                instruccion(key, instArray)
                break

        # Si no hay ninguna coincidencia, la cadena es inválida
        if find_match == False:
            print(f"[ERROR] La siguiente expresión en la línea {ln_num} NO es válida: {input_str}")

        ln_num += 1


# Función para ejecutar una instrucción en el Canvas de dibujo 'Py-cairo'
def instruccion(key, instArray):
    global angulo
    global color
    global radio
    global pos

    if key == "adelante":
        cr.set_source_rgba(color[0],color[1],color[2],color[3]) #Selecciona el color de la pluma

        radio *= int(instArray[1])                              #Elige cuanto se va a desplazar

        cr.rel_line_to(                                         #Se desplaza de acuerdo al radio y ángulo establecidos globalmente
            radio*math.cos(angulo),
            -radio*math.sin(angulo)
        )

        pos_temp = cr.get_current_point()                       #Modifica la posición de la pluma global
        pos[0] = pos_temp[0]
        pos[1] = pos_temp[1]

        cr.stroke()                                             #Guarda el dibujo (Esto hace que olvide la posición de la pluma)
        cr.move_to(                                             #"Recuerda" la posición de la pluma
            pos[0],
            pos[1]
        )
        radio = 100                                             # Regresa al valor 'default' del radio

    elif key == "atras":
        cr.set_source_rgba(color[0],color[1],color[2],color[3]) #Selecciona el color de la pluma

        radio *= int(instArray[1])                              #Elige cuanto se va a desplazar
        angulo += math.pi                                       #El ángulo da una media vuelta (180° = pi radianes)


        cr.rel_line_to(                                         #Se desplaza de acuerdo al radio y ángulo establecidos globalmente
            radio*math.cos(angulo),
            radio*math.sin(angulo)
        )

        pos_temp = cr.get_current_point()                       #Modifica la posición de la pluma global
        pos[0] = pos_temp[0]
        pos[1] = pos_temp[1]

        cr.stroke()                                             #Guarda el dibujo (Esto hace que olvide la posición de la pluma)
        cr.move_to(                                             #"Recuerda" la posición de la pluma
            pos[0],
            pos[1]
        )
        radio = 100                                             # Regresa al valor 'default' del radio

    elif key == "girarIzquierda":
        angulo += int(instArray[1])*math.pi/180                 # Modifica el ángulo de la pluma (En radianes)

    elif key == "girarDerecha":
        angulo -= int(instArray[1])*math.pi/180                 # Modifica el ángulo de la pluma (En radianes)

    elif key == "alzarPluma":
        color[3] = 0

    elif key == "bajarPluma":
        color[3] = 1

    elif key == "cambiarColor":
        cambiarColor(instArray[1])                              #Llama a la función 'cambiarColor()'

    elif key == "limpiarPantalla":
        cr.set_source_rgba(1,1,1,1)                             #Vuelve a crear el espacio de dibujo
        cr.rectangle(
            0,0,
            WIDTH,HEIGHT
        )
        cr.fill()

        cr.set_source_rgba(color[0],color[1],color[2],color[3])

    elif key == "centrarPluma":
        cr.move_to(
            WIDTH/2,
            HEIGHT/2
        )
```

```
    elif key == "repite":
        repetir(instArray)


#Función para cambiar el color de la pluma
def cambiarColor(colorSel1):
    global color
    colorSel = str(colorSel1)

    if colorSel == "ROJO" or colorSel =="rojo":
        color[0] = 0.8235
        color[1] = 0.1216
        color[2] = 0.2353

    elif colorSel == "NARANJA" or colorSel =="naranja":
        color[0] = 0.9922
        color[1] = 0.4039
        color[2] = 0.2275

    elif colorSel == "AMARILLO" or colorSel =="amarillo":
        color[0] = 0.9882
        color[1] = 0.9098
        color[2] = 0.5137

    elif colorSel == "VERDE" or colorSel =="verde":
        color[0] = 0.3137
        color[1] = 0.7843
        color[2] = 0.4706

    elif colorSel == "AZUL" or colorSel =="azul":
        color[0] = 0
        color[1] = 0.3098
        color[2] = 0.5961

    elif colorSel == "MORADO" or colorSel =="morado":
        color[0] = 0.3176
        color[1] = 0.1569
        color[2] = 0.5333

    elif colorSel == "NEGRO" or colorSel =="negro":
        color[0] = 0
        color[1] = 0
        color[2] = 0


# Función para ejecutar una instrucción del tipo 'repetir'
def repetir(instArray):
    global iteracion
    global iteracionmax
    iteracion += 1

        #Limpia la entrada de la instrucción 'repetir', específicamente del primer elemento y último debido a la estructura:
        #{instruccion1, instruccion2, ..., instruccionn} -> instArray = ['{instruccion1', 'instruccion2', '...', 'instruccionn}']
    if iteracion == 1:
        print("[WARNING]: La función 'repetir' está 'In-Progress', por lo que puede presentar fallos\n")
        indexTemp = instArray[1]
        indexTempArray = indexTemp.split("{")
        instArray[1] = indexTempArray[1]
        indexTemp = instArray[-2]
        indexTempArray = indexTemp.split("}")
        instArray[-2] = indexTempArray[0]

        #Debido a la estructura de instArray, debemos volver a construir las instrucciones y juntarlas con su parámatro
        #Ej: ['ADELANTE', '90'] -> ['ADELANTE 90']
        i = 0
        for element in instArray:
            if(element == "ADELANTE" or
            element == "adelante" or
            element == "ATRAS" or
            element == "atras" or
            element == "GIRA_I" or
            element == "gira_i" or
            element == "GIRA_D" or
            element == "gira_d" or
            element == "COLOR" or
            element == "color"):
                instArray[i] += ''.join(" " + instArray[i+1])
                del instArray[i+1]
            i += 1

        #Aqui se etxtrae la cantidad de repeticiones del último elemento en el array: [-1]: "([0-9])"
        indexTemp = instArray[-1]
        indexTemp = list(indexTemp)
        if (int(indexTemp[1]) <= 9):
            iteracionmax = int(indexTemp[1])
        else:
            print("Ingresaste un número inválido de repeticiones. El máximo es: 9")

        #Borra el elemento [0]: "repetir" & [-1]: "([0-9])" de la lista de instrucciones
        instArray.pop(0)
        instArray.pop(-1)

    print(f"============== Esta es la {iteracion}° iteración ==============\n")
```

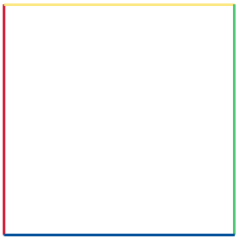
```
    ln_num1 = 1
    for element in instArray:
        for key, value in BNF.items():
            # Buscar una coincidencia de expresión regular
            match = re.match(value, element)

            # Si hay una coincidencia, la cadena es válida y llama a la función 'instruccion()'
            if match:
                print(f"La expresión en la línea {ln_num1} es válida. Gramática reconocida: '{key}'\n")
                instArray1 = element.split()
                instruccion(key, instArray1)
                break

        ln_num1 += 1
    if(iteracion < iteracionmax):
        repetir(instArray)

# Función 'main' con la impresión de la pantalla de Bienvenida
def main():
    print("\nBienvenido al minilenguaje LOGO 'i-script326'")
    print("Creado por: Ian Joab Padrón Corona\n")
    print("El programa lee un archivo de texto llamado 'instrucciones.txt'")
    print("que deberá estar ubicado en la misma carpeta de donde se está ejecutando este programa.\n")
    print("En ese archivo, podrás enscribir un conjunto de instrucciones para dibujar de acuerdo a la siguiente lista:\n")
    print("=============================================================================================================")
    print("|                                    -LISTA DE INSTRUCCIONES VALIDAS-                                       |")
    print("| ADELANTE/adelante [0-9]: Dibuja hacia adelante una linea de tamaña definido entre el 0-9.  Ej: ADELANTE 7  |")
    print("|                                                                                                           |")
    print("| ATRAS/atras [0-9]:       Dibuja hacia atras una linea de tamaña definido entre el 0-9.     Ej: atras 5     |")
    print("|                                                                                                           |")
    print("| GIRA_I/gira_i [1-359]:   Gira hacia la izquierda la cantidad de grados indicada para       Ej: GIRA_I 30   |")
    print("|                          dibujar en esa dirección.                                                        |")
    print("|                                                                                                           |")
    print("| GIRA_D/gira_d [1-359]:   Gira hacia la derecha la cantidad de grados indicada para         Ej: gira_d 60   |")
    print("|                          dibujar en esa dirección.                                                        |")
    print("|                                                                                                           |")
    print("| DIBUJAR/dibujar:         Baja la pluma para dibujar (Por defecto ya lo está)               Ej: DIBUJAR     |")
    print("|                                                                                                           |")
    print("| ALZAR/alzar:             Sube la pluma para dejar de dibujar                               Ej: alzar       |")
    print("|                                                                                                           |")
    print("| COLOR/color:             Cambia el color de la pluma                                       Ej: COLOR rojo  |")
    print("|     Hay 7 colores predefinidos:                                                                           |")
    print("|        ROJO,NARANJA,AMARILLO,VERDE,AZUL,MORADO,NEGRO                                                     |")
    print("|                                                                                                           |")
    print("| LIMPIAR/limpiar:         Borra todos los cambios hechos en el dibujo                       Ej: limpiar     |")
    print("|                                                                                                           |")
    print("| CENTRO/centro:           Regresa el lapiz al centro de la pantalla                         Ej: CENTRO      |")
    print("|                                                                                                           |")
    print("| REPITE/repite :          Ejecuta lo que esté dentro de las llaves { } la cantidad de                      |")
    print("| {<instrucciones>}        veces establecida en entre los paréntesis ( )                                    |")
    print("| (1-9)                                                                  Ej: repite {adelante 5 gira_i 90} (4)  |")
    print("|                                                                        <Esto creará un cuadrado ;)>       |")
    print("=============================================================================================================")
    print("Recuerda que SOLO podrás poner UNA instrucción POR línea. Diviertete :D\n\n")
    validar_entrada()

#La función 'main' es la primera en ser llamada
main()

#Guardar el dibujo en un archivo con extensión .png
ims.write_to_png(f"./{FILE_NAME[:-3]}.png")

print("=============================================================================================================")
print("He terminado de dibujar. Si no ves ningún mensaje tipo:  ")
print("'[ERROR] La siguiente expresión en la línea x NO es válida'")
print("\nEntonces podrás encontrar tu dibujo en la misma carpeta de donde ejecutaste este programa")
print("con el nombre: 'miniLOGO.png'")
print("\nAtt. ~Ian326 (El Creador)\n\n")
```

Ejemplos de Uso

Ejemplo 1 Cuadrado de varios colores:

```
COLOR ROJO
ADELANTE 9
GIRA_D 90
COLOR AMARILLO
ADELANTE 9
GIRA_D 90
COLOR VERDE
ADELANTE 9
GIRA_D 90
COLOR AZUL
ADELANTE 9
GIRA_D 90
```

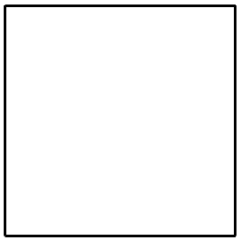
Resultado:



Ejemplo 2 Cuadrado con la instrucción ‘repite’:

```
REPITE {ADELANTE 9 GIRA_D 90} (2)
```

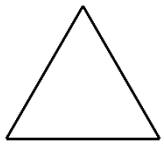
Resultado:



Ejemplo 3 Triángulo isósceles:

```
GIRA_I 90
ADELANTE 3
GIRA_D 120
ADELANTE 6
GIRA_D 120
ADELANTE 6
GIRA_D 120
ADELANTE 3
```

Resultado:



Ejemplo 4 Figura Abstracta:

```
ALZAR
ADELANTE 6
COLOR ROJO
DIBUJAR
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR AMARILLO
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR verde
GIRA_D 110
ADELANTE 6
GIRA_D 110
```

```
ADELANTE 6
COLOR AZUL
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR MORADO
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR NARANJA
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR ROJO
```

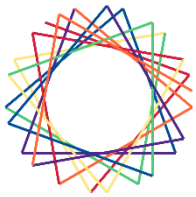
```
DIBUJAR
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR AMARILLO
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR verde
GIRA_D 110
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR AZUL
GIRA_D 110
```

```
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR MORADO
GIRA_D 110
```

```
ADELANTE 6
GIRA_D 110
ADELANTE 6
COLOR NARANJA
GIRA_D 110
```

```
ADELANTE 6
GIRA_D 110
ADELANTE 6
```

Resultado:



Experiencia de Aprendizaje

Gracias a esta entrega, puedo decir que los conceptos aprendidos en el periodo son de gran ayuda para futuros proyectos en programación y desarrollo de software. Por ejemplo, el léxico es un conjunto de palabras utilizadas en un lenguaje de programación y se requiere de su entendimiento para comprender efectivamente el código y los programas. Los autómatas son una herramienta para representar el comportamiento de un sistema o programa y son esenciales para el análisis y diseño de sistemas complejos. Los diagramas de sintaxis, a su vez, son una forma gráfica de mostrar la estructura de un programa y son útiles para comprender su lógica. Finalmente, las gramáticas BNF se utilizan para definir la estructura de un lenguaje de programación y necesaria para poder diseñar y crear lenguajes de programación.

En resumen, he aprendido a programar de manera efectiva y comprender cómo funcionan los lenguajes de programación. Esto a través de distintas herramientas que permiten analizar, diseñar y construir programas como lo fue mi mini lenguaje LOGO y alcanzar la expectativa de las competencias en esta parte del curso.

Video

https://drive.google.com/drive/folders/1cJn8w2jg-6h9PAVhJNGuI4h3D30tbp9l?usp=share_link