Tecnológico de Monterrey

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

Diego Andrés Figueroa Peart
A01660987

Curso:
Implementación de métodos computacionales
(TC2037.820)

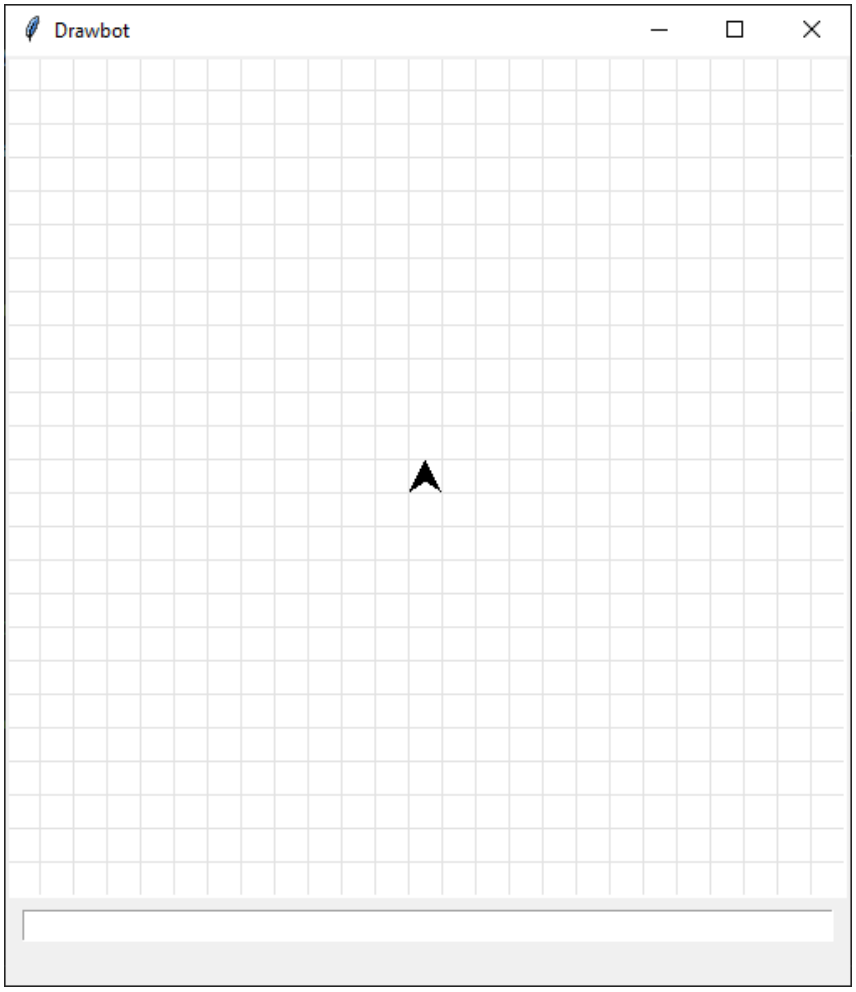
Profesores:
Román Martínez Martínez
Alberto Oliart Ros
Valentina Narváez Terán
Germán Domínguez Solís

Fecha de entrega:
20 de marzo de 2023

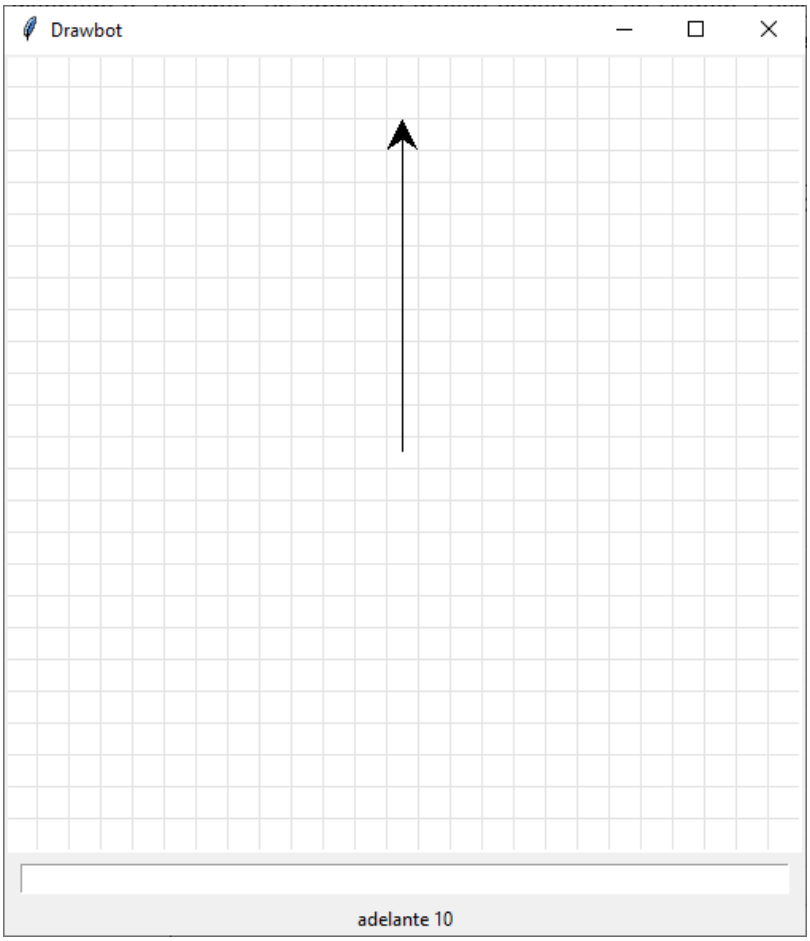
Evidencia #1: Drawbot

La situación problema del primer periodo de esta materia consistió en la creación de un lenguaje de programación de dominio específico (DSL) inspirado en Logo, un lenguaje simple que consiste en un ambiente gráfico con una tortuga que dibuja en pantalla y cuyo movimiento puede ser controlado con comandos básicos. Para este fin, he diseñado a “Drawbot” un ambiente gráfico construido en Python 3 utilizando el paquete Tkinter, que consiste en una flechita que, al igual que la tortuga de Logo, dibuja en pantalla y se mueve mediante comandos escritos por el usuario o cargados en un script.

La interfaz gráfica inicial de Drawbot es la siguiente:



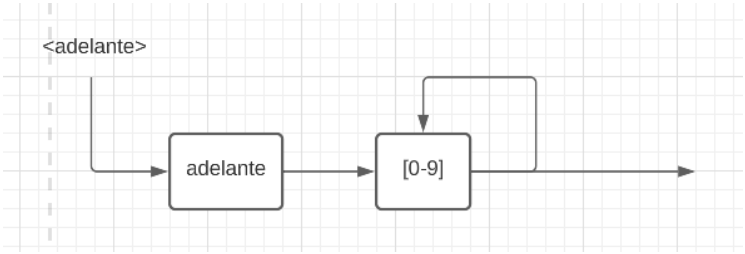
El robot se encuentra dentro de una cuadrícula (cuyo tamaño y número de celdas puede ser personalizado) y al moverse deja un rastro que sirve para hacer dibujos simples. Los comandos se pueden ingresar en la caja de texto en la parte inferior.



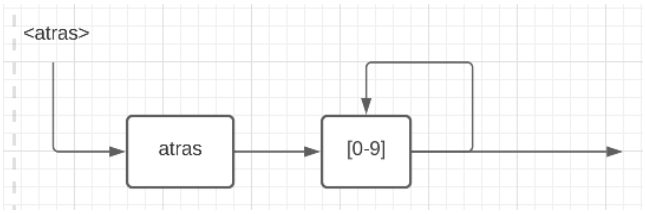
Como es posible observar, el comando más reciente es desplegado en pantalla debajo de la caja de entrada; si la entrada provocó un error, este también será desplegado en esta sección.

Los comandos o lexemas que acepta Drawbot son los siguientes:

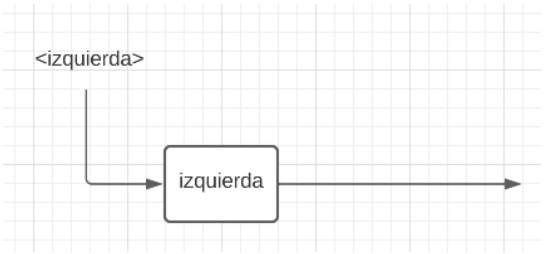
- **adelante n –** Drawbot se moverá hacia adelante n número de casillas.
    - **Expresión regular:** /adelante [0-9]+/
    - **Diagrama de sintaxis:**



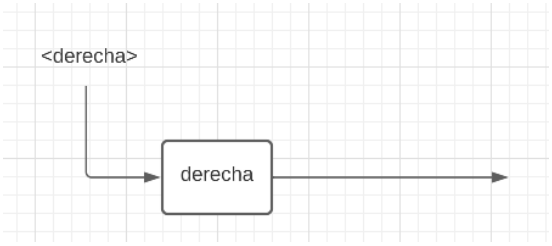
- **atras n –** Drawbot se moverá hacia atrás n número de casillas.
    - **Expresión regular:** /atras [0-9]+/
    - **Diagrama de sintaxis:**



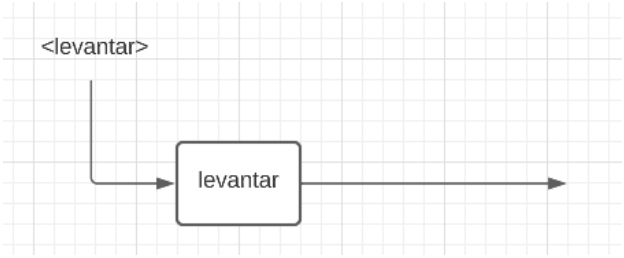
- **izquierda –** Drawbot rotará 90° a la izquierda.
    - **Expresión regular:** /izquierda/
    - **Diagrama de sintaxis:**



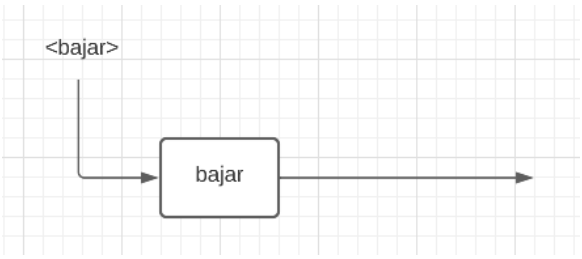
- **derecha –** Drawbot rotará 90° a la derecha.
    - **Expresión regular:** /derecha/
    - **Diagrama de sintaxis:**



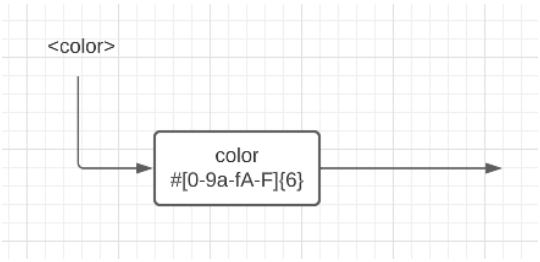
- **levantar –** Drawbot dejará de dibujar al moverse.
    - **Expresión regular:** /levantar/
    - **Diagrama de sintaxis:**



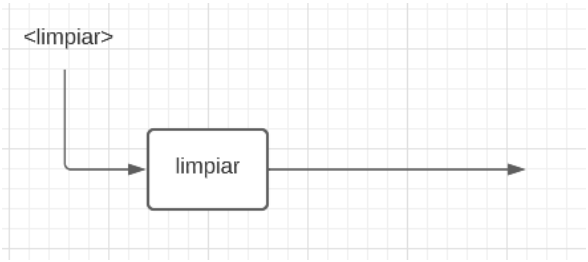
- **bajar –** Drawbot comenzará a dibujar al moverse.
    - **Expresión regular:** /bajar/
    - **Diagrama de sintaxis:**



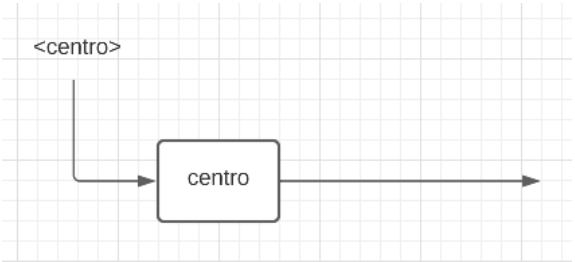
- **color #nnnnnn –** El color de la pluma cambiará al valor hexadecimal especificado.
    - **Expresión regular:** /color #[0-9a-fA-F]{6}/
    - **Diagrama de sintaxis:**



- **limpiar –** La pantalla se despejará de todos los dibujos.
    - **Expresión regular: /**limpiar/
    - **Diagrama de sintaxis:**

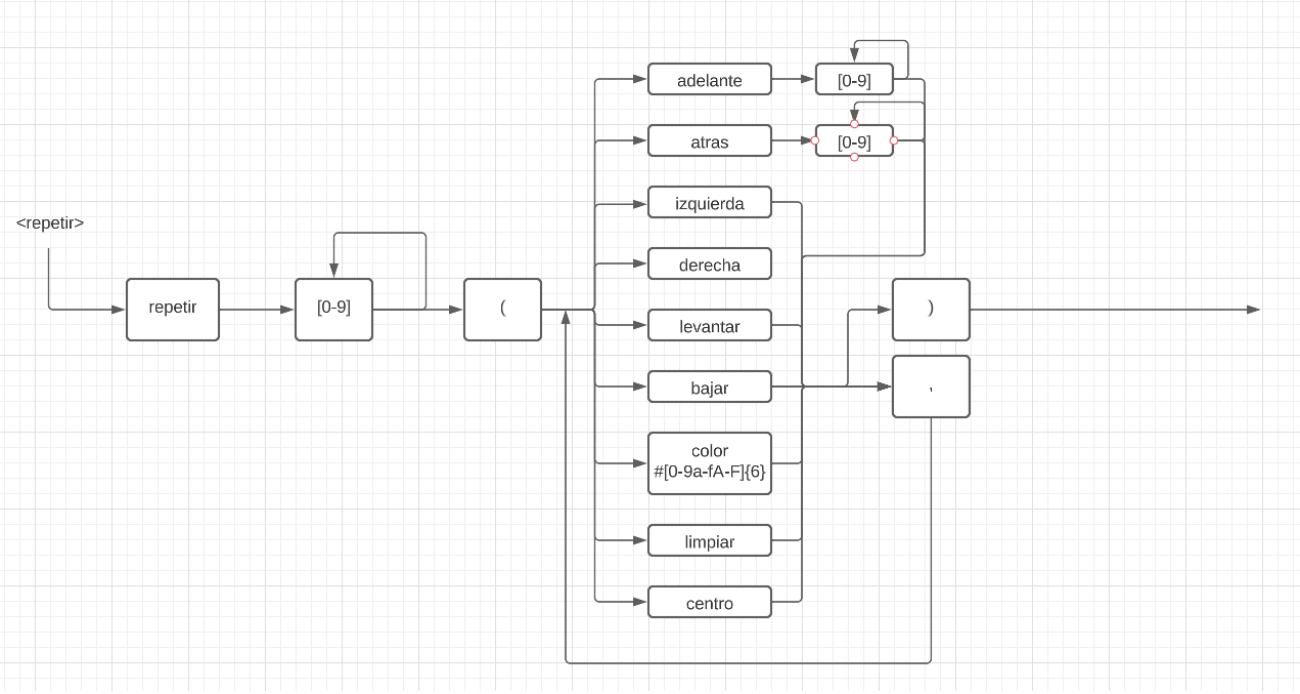


- **centro –** Drawbot regresará al centro exacto de la pantalla apuntando hacia arriba.
    - **Expresión regular: /**centro/
    - **Diagrama de sintaxis:**

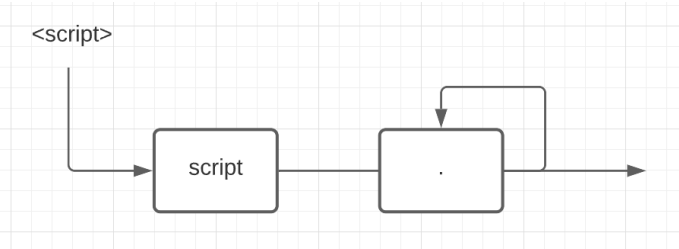


- **repetir n (comando1, comando2, …) –** Los comandos dentro de los paréntesis se repetirán en orden n número de veces.

- **Expresión regular:**
  /repetir [0-9]+ \((adelante [0-9]+|atras [0-9]+|izquierda|derecha|levantar|bajar|color #[0-9a-fAF]{6}|limpiar|centro|,)+\)/
- **Diagrama de sintaxis:**



- **script "archivo.txt" –** Se ejecutarán los comandos escritos línea por línea dentro del archivo txt especificado.
  - **Expresión regular:** /script .*/
  - **Diagrama de sintaxis:**

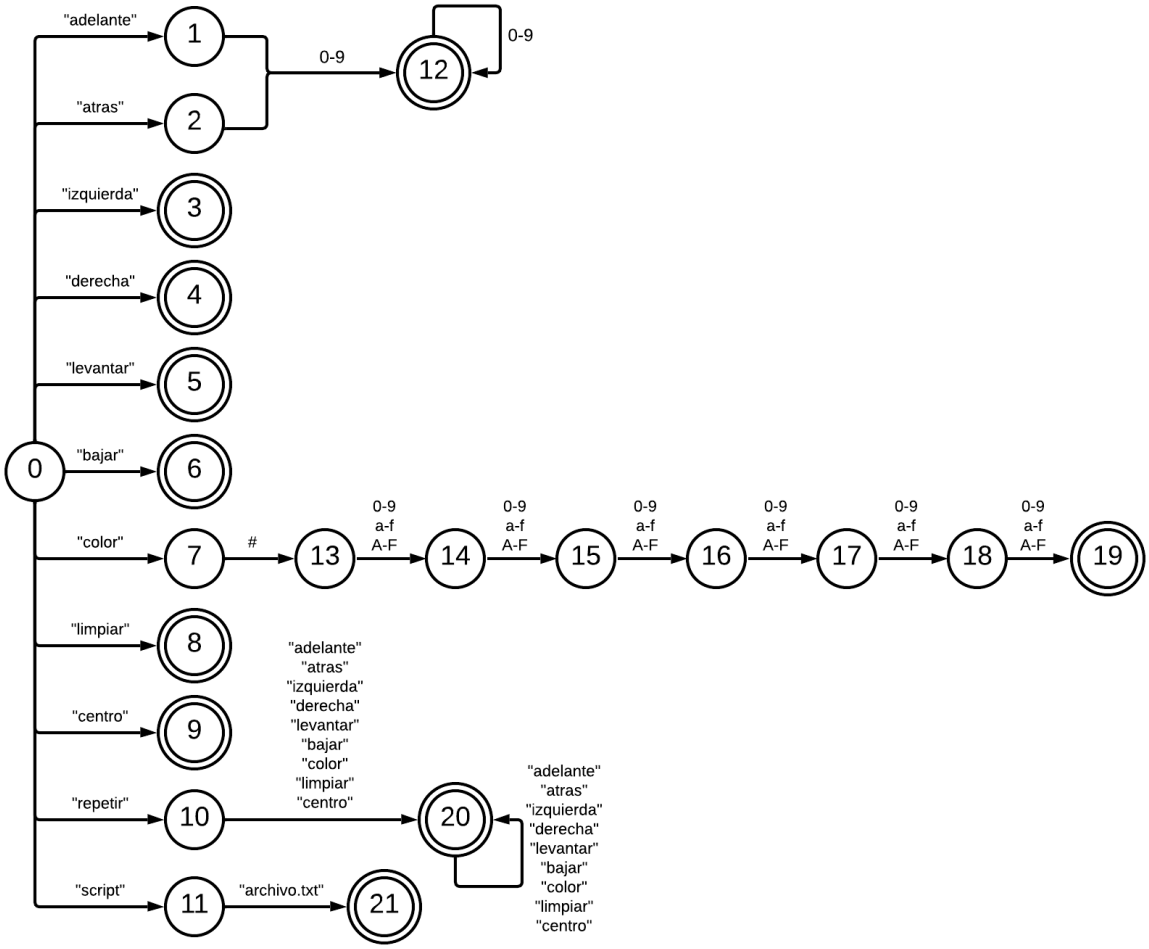


Gramática BNF:

Para comenzar a implementar el parser, primero diseñé la gramática del lenguaje utilizando el modelo EBNF:

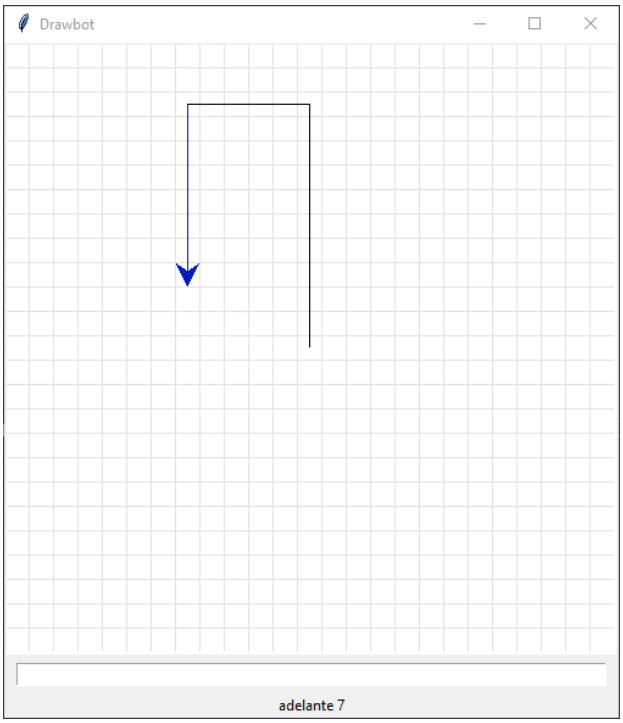
- `<comando> ::= <adelante> | <atras> | <izquierda> | <derecha> | <levantar> | <bajar> | <color> | <limpiar> | <centro> | <repetir> | <script>;`
- `<int> ::= (0 | 1 | 2 | 3 | 4 | 5| 6 | 7 | 8 | 9)*;`
- `<hex> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'A' | 'B' | 'C' | 'D' | 'E' | 'F';`
- `<adelante> ::= "adelante " <int>;`
- `<atras> ::= "atras " <int>;`
- `<izquierda> ::= "izquierda";`
- `<derecha> ::= "derecha";`
- `<levantar> ::= "levantar";`
- `<bajar> ::= "bajar";`
- `<color> ::= "color " '#' <hex> <hex> <hex> <hex> <hex> <hex>;`
- `<limpiar> ::= "limpiar";`
- `<centro> ::= "centro";`
- `<repetir> ::= "repetir " <int> '(' <comando> {<comando>} ')';`
- `<script> ::= "script " '"' {.} '"';`

El parser de Drawbot estará regido por el siguiente autómata determinístico:

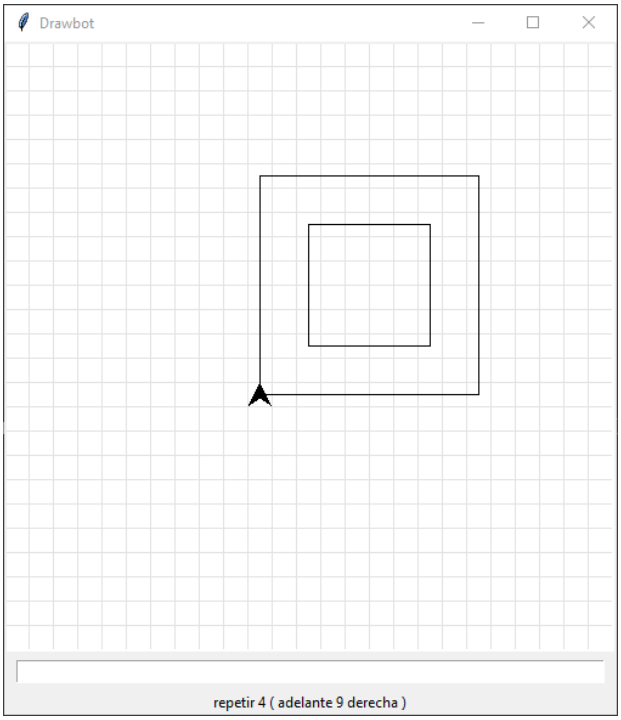


Pruebas:

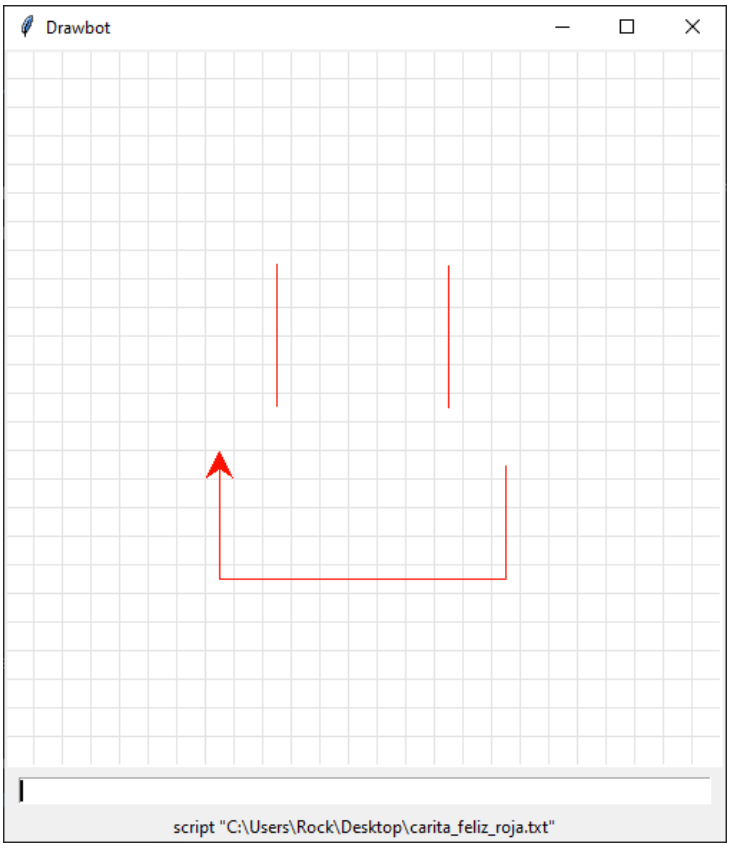
- Input:
  - adelante 10
  - derecha
  - atras 5
  - derecha
  - color #001fb8
  - adelante 7



- Input:
  - repetir 4 (adelante 5 derecha)
  - levantar
  - atras 2
  - izquierda
  - adelante 2
  - derecha
  - bajar
  - repetir 4 (adelante 9 derecha)



- Input: Un script llamado “carita_feliz_roja.txt” precargado con los siguientes comandos:
    - color #ff1100
    - levantar
    - izquierda
    - adelante 3
    - derecha
    - bajar
    - adelante 5
    - levantar
    - derecha
    - adelante 6
    - derecha
    - bajar
    - adelante 5
    - levantar
    - adelante 2
    - izquierda
    - adelante 2
    - derecha
    - bajar
    - adelante 4
    - derecha
    - adelante 10
    - derecha
    - adelante 4



Comentario acerca de los aprendizajes:

Personalmente, esta materia se me ha hecho bastante compleja ya que nunca había tocado los fundamentos del análisis léxico y sintáctico de los lenguajes de programación, sin embargo, rápidamente incrementó mi interés en estos temas, especialmente la implementación de un parser utilizando recursión. Este proyecto me pareció ideal para aplicar los aprendizajes obtenidos a lo largo de las 5 semanas que llevamos de la materia, me ayudó mucho para la comprensión de cuál es la manera correcta de analizar una entrada de usuario, como probar el código, y cómo arreglar los errores que se presentan durante el proceso de codificación. En general, me encantó este proyecto, y espero que los trabajos que llevemos en los siguientes periodos sean de igual calidad.