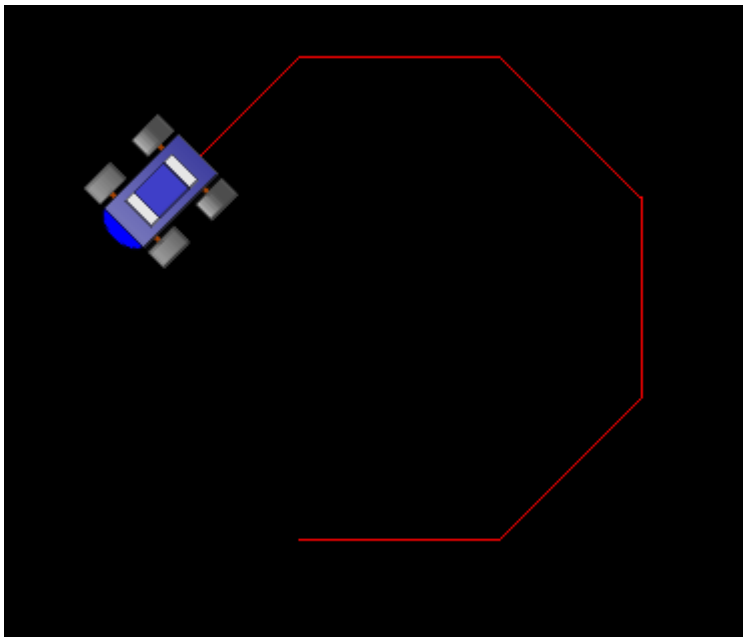
Lenguajes de Programación – Proyecto Semestral

El proyecto de este semestre consiste en implementar un intérprete para un lenguaje llamado CAR. La principal característica de este lenguaje es la capacidad de desplegar en pantalla un carro que puede dibujar líneas en la pantalla.

El carro puede moverse por la pantalla, rotar y dibujar con diferentes colores. Por ejemplo, el siguiente programa:

```
color 255,0,0,255
let i = 0
while (i < 5) {
      run_forward 100
      turn_left 45
      i = i + 1
}
run_forward 100
```

Produce la siguiente salida en pantalla:



Por supuesto, esto es sólo la punta del iceberg. Así como cualquier lenguaje tradicional de programación, CAR puede procesar expresiones aritméticas, lógicas, sentencias de control (condicionales, ciclos), funciones/subrutinas, etc.

Otro ejemplo que incluye la mayoría de los elementos del lenguaje:

```
function moveRotate(angle, dist)
{
      turn_right angle
      run_forward dist
}

function makeHorizontalLine()
{
      color 0,0,0,0
      run_forward 50
      turn_left 180
      color 255,255,255,255
      run_forward 100
      turn_left 180
      color 0,0,0,0
      run_forward 50
}

let a = 0
while (a < 3)
{
      writeln a
      a = a + 1
}
if (a==3)
{
      writeln "hola"
}
makeHorizontalLine()
color 255,255,255,255
turn_right 90
run_forward 200
moveRotate(45,50)
moveRotate(45,50)
moveRotate(45,50)
moveRotate(45,50)
```

Los elementos del lenguaje son los siguientes:

1. Comandos del carro: `run_forward`, `run_backwards`, `turn_left`, `turn_right`, `color`
2. Variables: declaración y asignación

   ```
   let miVariable
   miVariable = 5
   let miVariable2 = "hola"
   ```

3. Constantes y tipos de datos: Números decimales, booleanos y strings

   ```
   100.3
   #t
   ```

```
#f
"hola"
```

4. Condicionales

```
if (condición) {    }
if (condición) {    } else {  }
```

5. Ciclos

```
while (condición) {  }
```

6. Impresión y lectura por pantalla:

```
read miVariable
writeln miVariable + 4 *3
```

7. Expresiones aritméticas: `+`, `-`, `*`, `/`, inverso aditivo
8. Expresiones lógicas: `&&`, `||`, `!`
9. Expresiones de comparación: `>`, `<`, `<=`, `>=`, `==`, `<>`
10. Declaración de funciones

```
function miFuncion(param1, param2, … , paramN) {  }
```

11. Ejecución de funciones

```
miFuncion(10, “hola”, … , #t)
```

Para desarrollar el intérprete, Ud. dispondrá de las siguientes herramientas: Una distribución de Eclipse con soporte para ANTLR4 y un proyecto en Eclipse con la implementación básica del ambiente CAR (las librerías que muestran y operan el carro en la pantalla). Todos estos recursos serán publicados por Uvirtual.

Entregas

Primera Entrega:

•Intérprete básico de Car. Sólo comandos para mover el carro: `run_forward, run_backwards, turn_left, turn_right, color`

Segunda Entrega:

•Intérprete completo de Car (todos los elementos del lenguaje).