Elaborado por: Johan Alexis Bautista Quinayas: j.bautistag@uniandes.edu.co

18 de agosto de 2023
ISIS-1106

GUÍA DE APOYO P0 – LENGUAJES Y MÁQUINAS (5)

Aclaraciones con el enunciado:

El enunciado comprende el funcionamiento de un robot, con aspectos complejos que en este momento no son fundamentales para la entrega. En términos generales, el producto a entregar en esta entrega es un programa en Phyton o Java (sin uso de librerías externas, en especial Java cc no está permitido) que consiste en un parser que indique si una cadena dada cumple o no con las condiciones del lenguaje desarrollado en el enunciado.

En este sentido, es importante destacar que el entregable de esta entrega no comprende ninguna parte del funcionamiento del robot, sin embargo, es indispensable comprender este funcionamiento, se recomienda centrarse en entender las restricciones y gramática del lenguaje que se emplea para manejar el robot. Lo anterior inicia en la página 2:

Universidad de los Andes

ISIS-1106 Lenguajes y máquinas
Project 0
Fecha: August 13, 2023

The attached Java project includes a simple interpreter for the robot. The interpreter reads a sequence of instructions and executes them. An instruction is a command followed by ";".

A command can be any one of the following:

- M: to move forward
- R: to turn right
- C: to drop a chip
- B: to place a balloon
- c: to pickup a chip
- b: to grab a balloon
- P: to pop a balloon
- J(n): to jump forward n steps. It may jump over obstacles, but the final position should not have an obstacle.
- G(x,y): to go to position (x,y). Position (x,y) should not have an obstacle.

The interpreter controls the robot through the class `uniandes.lym.robot.kernel.RootWorldDec`

Figure 2 shows the robot before executing the commands that appear in the text box area at the bottom of the interface.

Figure 3 shows the robot after executing the aforementioned sequence of commands. The text area in the middle of the figure displays the commands executed by the robot.

Below we define a language for programs for the robot.

A program for the robot is simply a secuence of commands and definitions.

- A definition can be a variable defintion or a procedure definition.
  - A variable definition starts with the keyword `defVar` followed by a name followed by an initial value.
  - A procedure definition starts with the keyword `defProc` followed by by a name, followed by a list of parameter in parenthesis separated by commas, followed by a block of commands.
- A block of commands is a sequence of commands separated by semicolons within curly brackets .

Es importante entender este tipo de reglas de la gramática del robot.

De igual manera, las últimas tres páginas relatan más reglas que debe seguir el lenguaje del robot, las cuales son fundamentales para esta entrega. Por otro lado, las especificaciones de esta entrega se encuentran en la página 5 (Task 1).

Elaborado por: Johan Alexis Bautista Quinayas: j.bautistag@uniandes.edu.co

18 de agosto de 2023
ISIS-1106

Universidad de los Andes

ISIS-1106 Lenguajes y máquinas
Project 0
Fecha: August 13, 2023

Loop: `while condition Block` – Executes `Block` while `condition` is true.

RepeatTimes: `repeat v times Block` – Executes `Block` `n` times, where `v` is a value.

– A condition can be:

* `facing(D)` – where `D` is one of: north, south, east, or west
* `can(C)` – where C is a simple command. This condition is true when the simple command can be executed without error.
* `not: cond` – where `cond` is a condition

Spaces, newlines, and tabulators are separators and should be ignored.

The language is not case-sensitive. This is to say, it does not distinguish between upper and lower case letters.

Remember the robot cannot walk over obstacles, and when leaping it cannot land on an obstacle. The robot cannot walk off the boar or land off the board when leaping.

Task 1. The task of this project is to use Python or Java to implement a simple yes/no parser. The program should read a text file that contains a program for the robot, and verify whether the syntax is correct.

You must verify that used function names and variable names have been previously defined or in the case of functions, that they are the function's arguments. You must allow recursion.

Spaces and tabulators are separators and should be ignored.

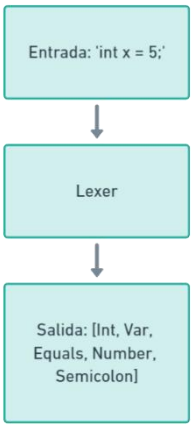
Below we show an example of a valid program.

Esto es lo que se debe entregar para el p0.

Recomendaciones:

La facilidad de implementación de un parser depende en parte de tu familiaridad con las bibliotecas y herramientas disponibles en cada lenguaje. Tanto Python como Java tienen sus ventajas y desventajas en términos de facilidad y estructura. La elección entre ellos dependerá de tus preferencias, sin embargo, existen algunas herramientas como las funciones de manejo de Strings en Python que podrían ser útiles.

Existen múltiples maneras de desarrollar un parser para una gramática de manera satisfactoria, recursividad, arreglos, etc. En este sentido, existe la herramienta lexer, que permite transformar palabras en tokens. Ejemplo:



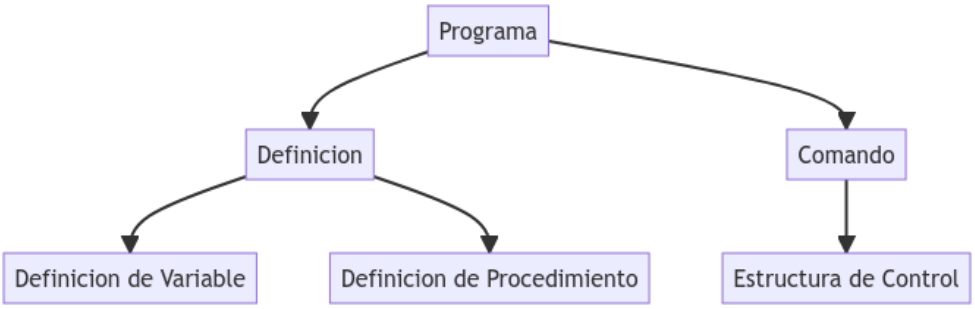

```python
def lexer(expression):
    tokens = []
    for char in expression:
        if char == '+':
            tokens.append(('PLUS', char))
        elif char == '-':
            tokens.append(('MINUS', char))
    return tokens

"""En este sencillo ejemplo, se itera a través
de cada carácter en la cadena de entrada y verifica
si es un operador + o -. Si lo es, agrega un token
correspondiente a la lista de tokens."""
```

Luego, es más fácil analizar una cadena o arreglo de tokens que un texto completo.

También se recomienda hacer una limpieza de la cadena original recibida por parámetro, eliminando espacios (no entre palabras), tabulaciones, manejando solo mayúsculas o minúsculas, etc. De igual manera, es más fácil trabajar sobre una cadena limpia y uniforme que sobre un texto original.

Por otro lado, es importante comprender la estructura de la gramática y cada una de sus reglas, es por esto que, puede ser útil plasmarla gráficamente para entender su complejidad y recursividad. Una buena señal para empezar a programar es saber qué estructuras, condiciones o declaraciones pueden ir después de cada una de x estructura, condición o declaración de la gramática del robot. Por ejemplo:



(Ejemplo no aplicable al escenario propuesto, solo tomar como referencia).

Motivación:

Este proyecto te brindará la libertad de elegir tecnologías, enfoques, así como las mejores estrategias a seguir, permitiéndote expandir tus habilidades. Al programar y comprender un parser, dominarás conceptos cruciales de programación, así como los conocimientos analíticos cruciales para resolución de problemas. Esta experiencia más allá de ser una asignación es una oportunidad de poner a prueba tus conocimientos, habilidades y tu propia creatividad.

- → El P0 no es simplemente una tarea académica; es una oportunidad única para sumergirse en el fascinante mundo de los lenguajes de programación y las máquinas. Algunas razones por las cuales es esencial abordar este proyecto con integridad y autenticidad son:
- → **Aprendizaje Real:** El verdadero aprendizaje ocurre cuando te enfrentas a desafíos y buscas soluciones por ti mismo. Al hacer trampa, te estás privando de una valiosa oportunidad de aprendizaje. Así como una **Preparación para Desafíos Futuros**, este proyecto es una preparación para desafíos más grandes que enfrentarás en el futuro. Al abordarlo con seriedad y compromiso, te estás preparando para superar obstáculos más grandes en el futuro.

Restricciones de entrega:

- → No se permite el uso de librerías externas, especialmente no se permite Java cc. No obstante, se pueden emplear bibliotecas nativas de cada lenguaje, por ejemplo: java.util, for with enumerate (phyton), etc.
- → Se debe adjuntar un documento pdf con una breve documentación acerca de cómo planteó la solución al problema propuesto. Por otro lado, si no se logra una solución satisfactoria, explicar brevemente qué errores tuvo y cómo se podrían corregir.
- → Adjuntar un archivo .zip nombrado con el formato Apellido(s)-Nombre(s)-P0. Por ejemplo: Bautista-Quinayas-Johan-Alexis-p0