DESIGN GRAMMAR

Docente:
Vicente Enrique Machaca Arceda

Integrantes:
Frank Jhoseph Duarte Oruro

Universidad La Salle
2022

1 Introducción

El lenguaje definido es un formato basado en líneas compactas que estas mismas pueden ser analizadas en una Analizador lexico, donde tenemos que especificar los caracteres y gramáticas para poder reconocer el lenguaje, entonces para poder reconocer ahora la gramatica se tiene que reutilizar los tokens utilizados antes, sin embargo ahora se quiere analizar el diseño de la gramatica donde se analizara la expresion por linea. Se tiene tambien ya reconocidos los caracteres que se utilizaran.

```
TrueLiteral ::= "verdad"
FalseLiteral ::= "falso"
NotExpression ::= "!" Expression
ArrayType ::= "entero" "[" "]"
Type ::= tipoArray
| TipoBoolean
| TipoInteger
| ID
BooleanType ::= "boolean"
IntegerType ::= "entero"
Statement ::= Block
| AssignmentStatement
| ArrayAsignamientoTestamento
| IfStatement
| WhileStatement
| PrintStatement
Block ::= "{" ( Testament)* "}"
AssignmentStatement ::= ID "=" Expression
ArrayAssignmentStatement ::= ID "[" Expression "]" "=" Expression
IfStatement ::= "si" "(" Expression ")" Statement "sino" Statement
WhileStatement ::= "mientras" "(" Expression ")" Statement
PrintStatement ::= "Imprimir" "(" Expression ")"
```

2 Ejercicio

Ahora se probara un codigo donde tenemos una gramatica de caracteres donde se reconocerea sus

```
ciudades = [’Berlin’, ’São Paulo’, ’Tokyo’, ’New York’]
capitales = [’Berlin’, ’Tokyo’]
para city en ciudades:

  si ciudad not en capitales:
    continuar
  Imprimir(f’{ciudad} is a capital’)
```

Veremos la gramatica del lenguaje utilizado, donde se vera las expresiones o caracteres analizados que pueden existir

```
single_input::= NUEVALINEA | SIMPLE_testament

eval_input::= Expression  "=" "["Expression ,Expression ,Expression ,Expression  "]
| " "

eval_input::= "["Expression ,Expression "]
| " "

ForStatement::="para" Expression  "en"  Expression
| " "

IfStatement::= "si" Expression "not" "en" Expression ":"
| " "

eval_input::= "continar" | " "

PrintStatement ::= "Imprimir""("Expression "{" Expression"}" Expression")"
| "f’{Expresion} is a Expression"
```