

## UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS FACULTAD DE INGENIERÍA

#### **Docente**

Canaval Sánchez, Luis Martin

Carrera: Ciencias de la Computación

Estudiante: Espíritu Cueva, Renzo Andree

Código: U202113340

# Índice

Código de Ejemplo Utilizado (DOT language)	3
Implementación del Parser	
Implementación de Lexer	
Arbol Sintactico de Abstracción (AST)	
• •	
Github (Dot Compiler)	c

## Código de Ejemplo Utilizado (DOT language)

```
digraph G {
 subgraph cluster_0 {
  style=filled;
  color=lightgrey;
  node [style=filled,color=white];
  a0 -> a1 -> a2 -> a3;
  label = "process #1";
 }
 subgraph cluster_1 {
  node [style=filled];
  b0 -> b1 -> b2 -> b3;
  label = "process #2";
  color=blue
 }
 start \rightarrow a0;
 start -> b0;
 a1 -> b3;
 b2 -> a3;
 a3 -> a0;
 a3 -> end;
 b3 -> end;
 start [shape=Mdiamond];
 end [shape=Msquare];
```

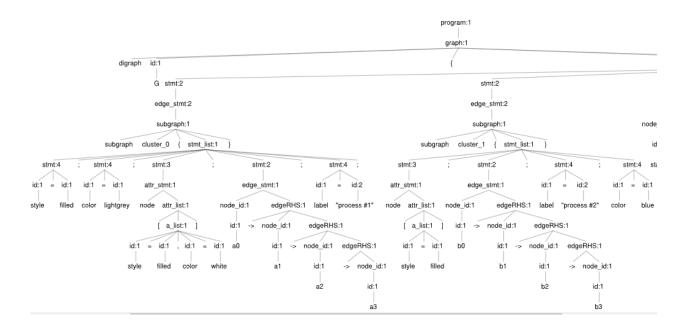
#### Implementación del Parser

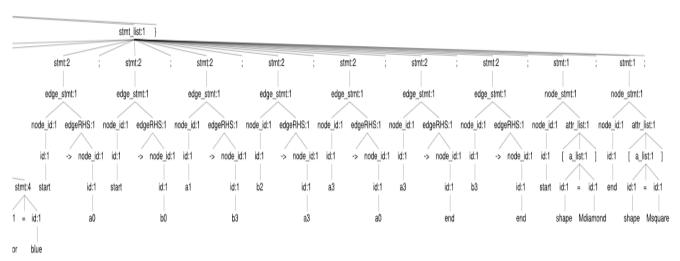
```
parser grammar ExprParser;
options { tokenVocab=ExprLexer; }
program
    : graph
graph
    : gd=(GRAPH|DIGRAPH) id '{ 'stmt_list '} ' #Grafo
stmt_list
    : ( stmt ';'? )*
                                  #Stc
stmt
    : node_stmt
                                    #Nodo
    | edge_stmt
                                    #Edge
    attr_stmt
                                  #Attr
    | id '=' id
                                 #Assing
attr_stmt
    : gne=(GRAPH| NODE| EDGE) attr_list
                                           #AList
attr_list
    : ('['a_list']')+
                                  #Aarr
a_list
    : (id ('=' id)+ ','?)+
                                   #Ass
edge_stmt
    : (node_id|subgraph) edgeRHS? attr_list?
                                              #Subgrafo
    : EDGEOP (node_id|subgraph) edgeRHS?
                                                  #NodeEdge
node_stmt
    : node_id attr_list
                                   #NodeAttr
node_id
    : id port?
                                  #NodeO
```

## Implementación de Lexer

```
lexer grammar ExprLexer;
GRAPH: 'graph';
DIGRAPH: 'digraph';
SUBGRAPH: 'subgraph';
NODE: 'node';
EDGE: 'edge';
EQUAL: '=';
POINT : ':';
LCURLY: '{';
RCURLY:'}';
LBRACKET: '[';
RBRACKET: ']';
SEMICOLON: ';';
ENDC: ',';
EDGEOP: '->'|'--';
ID: [a-zA-Z_][a-zA-Z_0-9]*;
STRING: "" ('\\"'|.)*? "";
NUMBER: '-'? ('.' [0-9]+|[0-9]+ ('.' [0-9]*)?);
WS: [ t / r / f] + -> skip ;
```

### Arbol Sintactico de Abstracción (AST)





### **Github (Dot Compiler)**

https://github.com/Raichi1/DotCompiler.git