



Patito-Entrega #1

Rodrigo Merino de la Parra A00836396

Expresiones Regulares

```
ID = ^[A-Za-z][A-Za-z0-9_]*$
```

```
CTE_INT = ^[+|-]?\d+$
```

```
CTE_FLOAT = ^[+|-]?\d+\.\d+$
```

```
CTE_STRING = ^"(?:\\.|[^\"])*"$
```

Patito

```
<Programa>  
<Body>  
<ASSIGN>  
<CTE>  
<FUNCS>  
<EXPRESSION>  
<STATEMENT>  
<EXP>  
<TERMINO>  
<VARS>  
<PRINT>  
<CYCLE>  
<CONDITION>
```

<FACTOR>
<F_CALL>
<TYPE>

Lista de tokens

PROGRAM → "program"
VAR → "var"
INT → "int"
FLOAT → "float"
VOID → "void"
MAIN → "main"
IF → "if"
ELSE → "else"
WHILE → "while"
DO → "do"
PRINT → "print"
END → "end"

PLUS → "+"
MINUS → "-"
MULT → "*"
DIV → "/"

GT → ">"
LT → "<"
NEQ → "!="
EQ → "="

LPAREN → "("
RPAREN → ")"
LBRACE → "{"
RBRACE → "}"
LBRACKET → "["
RBRACKET → "]"

COMMA → ","

SEMICOLON → ";"

COLON → ":"

COMMENT_LINE → //[^\\n]*

COMMENT_BLOCK → /*([*]|*+[^*/])*+/*

WHITESPACE → [\\t\\r\\n]+

A → "A" a → "a"

B → "B" b → "b"

C → "C" c → "c"

D → "D" d → "d"

E → "E" e → "e"

F → "F" f → "f"

G → "G" g → "g"

H → "H" h → "h"

I → "I" i → "i"

J → "J" j → "j"

K → "K" k → "k"

L → "L" l → "l"

M → "M" m → "m"

N → "N" n → "n"

O → "O" o → "o"

P → "P" p → "p"

Q → "Q" q → "q"

R → "R" r → "r"

S → "S" s → "s"

T → "T" t → "t"

U → "U" u → "u"

V → "V" v → "v"

W → "W" w → "w"

X → "X" x → "x"

Y → "Y" y → "y"

Z → "Z" z → "z"

DIGIT_0 → "0"

DIGIT_1 → "1"

DIGIT_2 → "2"

DIGIT_3 → "3"

DIGIT_4 → "4"

DIGIT_5 → "5"

DIGIT_6 → "6"

DIGIT_7 → "7"

DIGIT_8 → "8"

DIGIT_9 → "9"

UNDERSCORE → "_"

QUOTE → "\""

DQUOTE → "\""

BACKSLASH → "\\"

AMPERSAND → "&"

PIPE → "|"

EXCLAM → "!"

QUESTION → "?"

PERCENT → "%"

CARET → "^"

TILDE → "~"

AT → "@"

HASHTAG → "#"

DOLLAR → "\$"

SPACE → " "

TAB → "\t"

NEWLINE → "\n"

RETURN → "\r"

EOF → fin de archivo

ERROR → cualquier carácter no reconocido

Gramatica libre de contexto

<FUNCS>

→ VOID

→ ID

→ ID

→ TYPE

<PRINT>

<PRINT> → PRINT' PRINT''
PRINT' → PRINT (EXPRESSION);

PRINT' → ~~PRINT~~ EXPRESSION PRINT''

PRINT' → CTE-STRING CTE-STRING PRINT''

PRINT'' → ε, PRINT'

PRINT'' → ε

<PROGRAMA>

<PROGRAMA> -

→ program id ; <program'> main BODY end

<program'> <VARs> <program'>

<program'> <VARs> / <program''>

<program''> <FUNCS> <program''>

<program''> <FUNCS> ε

→ VOID

DÍA / MES / AÑO

~~<FUNCS> → void id (<FUNCS'>) { <FUNCBODY> };~~

~~<FUNCS'> → id : <TUPE> <FUNCS''>~~

~~<FUNCS''> → ε~~

~~<FUNCS'''> → , id : <TUPE> <FUNCS''''>~~

~~<FUNCS''''> → ε~~

~~<FUNCS'''''> → <FUNCS''''''> <FUNCS''''''''>~~

~~<FUNCS''''''> → <FUNCS''''''''>~~

~~<FUNCS''''''''> → void id (<PARAMS>)~~

<FUNCS> → void id (<PARAMS>) <PARAMS'''>;

<PARAMS> → id : <TUPE> <PARAMS'>

<PARAMS> → ε

<PARAMS'> → , id : <TUPE> <PARAMS''>

<PARAMS'> → ε

<PARAMS''> → , id : <TUPE> <PARAMS'''>

<PARAMS'''> → <VARS> <BODY>

<PARAMS'''> → <BODY>

DÍA / MES / AÑO

<STATEMENT>

STATEMENT \rightarrow ASSIGN

STATEMENT \rightarrow CONDITION

STATEMENT \rightarrow CYCLE

STATEMENT \rightarrow F-CALL

STATEMENT \rightarrow PRINT

<TERMINO>

TERMINO \rightarrow FACTOR TERMINO'

TERMINO' \rightarrow • FACTOR TERMINO'

TERMINO' \rightarrow / FACTOR TERMINO'

TERMINO' \rightarrow ϵ

<BODY>

{ STATEMENTS

STATEMENT \rightarrow STATEMENT'

STATEMENT' \rightarrow ϵ

STATEMENT' \rightarrow STATEMENT

<VARS>

A = ID A'

B = A:TYPE ; B'

A' = A

B' = ϵ

A' = ϵ

B' = B

LOGRAND
ENTERTAINMENT GROUP

ENTERTAINMENT GROUP

<EXP>

$EXP \rightarrow \text{TERMINO } EXP'$

$EXP' \rightarrow + \text{TERMINO } EXP'$

$EXP' \rightarrow - \text{TERMINO } EXP'$

$EXP' \rightarrow \epsilon$

<CTE>

$CTE \rightarrow \text{CTE_INT}$

$CTE \rightarrow \text{CTE_FLOAT}$

<CYCLE>

$\text{while (EXPRESION) DO } \rightarrow \text{(VERBO);}$

<TYPE>

$\text{TUPE} \rightarrow \text{INT}$

$\text{TUPE} \rightarrow \text{FLOAT}$

<ASSIGN>

$\text{ASSIGN } \alpha \rightarrow \text{ID} = \text{EXP};$

$\langle F_CALL \rangle \rightarrow id(\langle Expression \rangle \langle expression' \rangle);$
 $\langle expression' \rangle \rightarrow \langle expression \rangle \langle expression' \rangle$
 $\langle expression' \rangle \rightarrow \epsilon$

DÍA / MES / AÑO

~~$\langle F_CALLS \rangle \rightarrow id(\langle Expression \rangle);$~~

~~$\langle Expression \rangle \rightarrow id(\langle Expression \rangle);$~~

~~$\langle F_CALL \rangle \rightarrow id(\langle Expression \rangle);$~~

~~$\langle Expression \rangle \rightarrow Expression^* Expression^*$~~

~~$\langle Expression \rangle \rightarrow \& Expression^*$~~

~~$\langle Expression \rangle \rightarrow \epsilon$~~

$\langle CONDITIONS \rangle$

$\langle CONDITIONS \rangle \rightarrow if(Expression) Body;$

$\langle CONDITIONS \rangle \rightarrow if(Expression) Body else Body;$

$\langle EXPRESSION \rangle$

$\langle EXPRESSION \rangle \rightarrow EXP EXP''$

$EXP'' \rightarrow > EXP''$

$EXP'' \rightarrow \langle EXP'' \rangle$

$EXP'' \rightarrow != EXP''$

$EXP'' \rightarrow \epsilon$



LOGRAND
ENTERTAINMENT GROUP

<FUNCS>

→ VOID

→ ID

→ ID

→ TYPE

<FACTOR>

<FACTOR> → (EXPRESSION)

~~<FACTOR> → <FACTOR> * <FACTOR>~~

~~<FACTOR> → <FACTOR> / <FACTOR>~~

~~<FACTOR> → <FACTOR> - <FACTOR>~~

~~<FACTOR> → <FACTOR> + <FACTOR>~~

~~<FACTOR> → - <FACTOR>~~

~~<FACTOR> → <CTE>~~

<FACTOR> → ID

<FACTOR> → (EXPRESSION)

<FACTOR> → <FACTOR'> <FACTOR''> <FACTOR'''>

<FACTOR''> → +

<FACTOR''> → -

<FACTOR''> → *

<FACTOR''> → /

<FACTOR''> → <CTE>

<FACTOR'> → <FACTOR''> <FACTOR'''>

Genera, como parte de la entrega, una documentación que describa brevemente los principales hallazgos del análisis de las diferentes herramientas. Agrega, a la documentación previa, cómo (en qué formato) diste de alta las reglas de construcción de Patito que contenga la definición de las expresiones regulares y reglas gramaticales desarrolladas. Agrega los principales Test-cases desarrollados para validar su funcionamiento. Considera que este documento irá creciendo conforme trabajes en las siguientes entregas.

****Tests del Lexer:****

cd tests → python3|python test_lexer.py

- 1 Reconocimiento de palabras reservadas
- 2 Validación de identificadores con letras, números y guiones bajos
- 3 Reconocimiento de constantes **enteras** (positivas, negativas)
- 4 Reconocimiento de constantes flotantes con signo
- 5 Manejo de cadenas literales entre comillas
- 6 Operadores aritméticos y comparación
- 7 Delimitadores y **puntuación** (paréntesis, llaves, corchetes, comas, etc.)
- 8 Ignorar comentarios de línea y de bloque

****Tests del Parser:****

cd tests → python3|python test_parser.py

- 1 Estructura mínima de programa válido
- 2 Declaraciones de variables múltiples con tipos
- 3 Asignaciones simples
- 4 Expresiones aritméticas con operadores múltiples
- 5 Expresiones de comparación
- 6 Sentencias **if** simples
- 7 Sentencias **if-else** completas
- 8 Ciclos **while-do**
- 9 Sentencias print con múltiples argumentos
- 10 Declaración de funciones con parámetros
- 11 Llamadas a funciones con argumentos
- 12 Expresiones complejas con precedencia y paréntesis
- 13 Detección de errores **sintácticos** (ej. punto y coma faltante)
- 14 Programa completo integrando todas las características