**Diseño de compliadores**

**Analizador Léxico**

**Daniel Charua A01017419**

**23/02/19**

<u>Expresiones regulares</u>

Se implementó del analizador léxico con expresiones regulares, los siguientes expresiones fueron usadas:

(r'[ \s\n\t]+',   TokenType.SPACE),

(r'',            TokenType.SPACE),

(r'#[^\n]*',     TokenType.SPACE),

(r'\$',          TokenType.ENDFILE),

(r'\{',          TokenType.SPECIAL),

(r'\}',          TokenType.SPECIAL),

(r'\[',          TokenType.SPECIAL),

(r'\]',          TokenType.SPECIAL),

(r'\(',          TokenType.SPECIAL),

(r'\)',          TokenType.SPECIAL),

(r'\/\*(\*(?!\/)|[^*])*\*\/', TokenType.COMMENT),

(r';',           TokenType.SPECIAL),

(r',',           TokenType.SPECIAL),

(r'\+',          TokenType.SPECIAL),

(r'-',           TokenType.SPECIAL),

(r'\*',          TokenType.SPECIAL),

(r'/',           TokenType.SPECIAL),

(r'<=',          TokenType.SPECIAL),

(r'<',           TokenType.SPECIAL),

(r'>=',          TokenType.SPECIAL),

(r'>',           TokenType.SPECIAL),

```
(r'==',          TokenType.SPECIAL),

(r'!=',          TokenType.SPECIAL),

(r'=',           TokenType.SPECIAL),

(r'else',        TokenType.RESERVED),

(r'if',          TokenType.RESERVED),

(r'int',         TokenType.RESERVED),

(r'return',      TokenType.RESERVED),

(r'void',        TokenType.RESERVED),

(r'while',       TokenType.RESERVED),

(r'[0-9]+[a-zA-Z]+', TokenType.ERROR),

(r'[0-9]+',      TokenType.NUM),

(r'[a-zA-Z_][0-9a-zA-Z_]*', TokenType.ID)]
```

Automata

**State diagram / DFA:**

- Start → State 1
- State 1: self-loop labeled `/n /s`
- State 1 → State 2: `0-9`
- State 2: self-loop labeled `0-9`
- State 2 → State 6: `/s`
- State 1 → State 3: `[+ - / * > < ! = ; , [ ] { }]`
- State 3 → State 6: `/s`
- State 1 → State 4: `[A-Z a-z]`
- State 4: self-loop labeled `[A-Z a-z 0-9 _]`
- State 4 → State 6: `/s`
- State 1 → State 5: `[if, else, int, return, void, while]`
- State 5 → State 6: `/s`