# Analisador léxico con Flex

Christofer Fabián Chávez Carazas
Universidad Nacional de San Agustín de Arequipa
Escuela Profesional de Ciencia de la Computación
Compiladores

9 de noviembre de 2017

## Problema **Desarrollar un scanner que reconozca lo siguiente:**

- ID
- NUM
- $+$, $-$, $*$, $/$
- $>$, $>=$, $<$, $>=$, $=$, $==$, $!=$
- {, }, [, ], ”, ;, (, )
- Ignorar Comentario de línea y de bloque.

## Programa

Las reglas se encuentran ordenadas. Al comienzo están las reglas para que el scanner ignore los comentarios, luego están las reglas que se comen los espacios, saltos de línea y las tabulaciones, luego se encuentran las reglas para las palabras reservadas (if - while), luego vienen los operadores, luego los delimitadores, luego están las reglas que reconocen un número y un identificador, finalmente está una regla por si se ingresa un caracter o cadena desconocida (para manejar errores).

```
%%

\/\/.+\n {
    printf("");
}

\/\*(.\n?)+\*\/ {
    printf("");
}

\n {
    printf("");
}

\t {
    printf("");
}

[ ] {
    printf("");
}
```

```
while {
    printf("WHILE-> %s\n", yytext);
}
if {
    printf("IF-> %s\n", yytext);
}



\+ {
    printf("MAS-> %s\n", yytext);
}
- {
    printf("MENOS-> %s\n", yytext);
}
\* {
    printf("MULT-> %s\n", yytext);
}
\/ {
    printf("DIV-> %s\n", yytext);
}


>= {
    printf("MAYOR IGUAL-> %s\n", yytext);
}
> {
    printf("MAYOR-> %s\n", yytext);
}
\<= {
    printf("MENOR IGUAL-> %s\n", yytext);
}
\< {
    printf("MENOR-> %s\n", yytext);
}
== {
    printf("IGUAL-> %s\n", yytext);
}
= {
    printf("ASIG-> %s\n", yytext);
}
!= {
    printf("DIF-> %s\n", yytext);
}
! {
    printf("NEG-> %s\n", yytext);
}



\{ {
    printf("LLAVE IZQ-> %s\n", yytext);
}

\} {
    printf("LLAVE DER-> %s\n", yytext);
}
\[ {
    printf("COR IZQ-> %s\n", yytext);
}
\] {
    printf("COR DER-> %s\n", yytext);
}
\( {
    printf("PAR IZQ-> %s\n", yytext);
}
\) {
    printf("PAR DER-> %s\n", yytext);
}
\" {
    printf("COMILLAS-> %s\n", yytext);
}



; {
    printf("PUNTO Y COMA-> %s\n", yytext);
}

[0-9]+ {
    printf("NUM-> %s\n", yytext);
}
[a-zA-Z]+[0-9]* {
    printf("ID-> %s\n", yytext);
}

. {
    printf("NO RECONOCIDO-> %s\n", yytext);
}

%%
```

```
main(){
    yylex();
}
```

## Experimentos y Resultados

```
xnpio@xnpio-Satellite-U40t-A:~/Documentos/Xnpio/UNSACS/Compiladores/Lab/Tarea6$ cat test
/*Archivo para testear
Estas lineas son pruebas
para el comentario en bloque*/
num1 = 10; //Asignacion
if(num1 == 10){ //If y igualdad
        num1 = 3 + 10 - 2; //Suma y resta
}
while(num1 >= 10){ //While y mayor igual
        num2 = num1 / 2; //Division
        if(num1 > 10) num1 = 9; //Mayor
        if(num2 <= num1) num2 = num1; //Menor Igual
        num2 = num1 * 3; //Multiplicacion
}
if(num2 != 10) num2 = 10; //Diferente
if(num1 < num2) num1 = num2; //Menor
num3[2] = num2; //Corchetes
xnpio@xnpio-Satellite-U40t-A:~/Documentos/Xnpio/UNSACS/Compiladores/Lab/Tarea6$
```

Figura 1: Archivo de prueba

```
xnpio@xnpio-Satellite-U40t-A:~/Documentos/Xnpio/UNSACS/Compiladores/Lab/Tarea6$ ./run < test
ID->num1
ASIG->=
NUM->10
PUNTO Y COMA->;
IF->if
PAR IZQ->(
ID->num1
IGUAL->==
NUM->10
PAR DER->)
LLAVE IZQ->{
ID->num1
ASIG->=
NUM->3
MAS->+
NUM->10
MENOS->-
NUM->2
PUNTO Y COMA->;
LLAVE DER->}
WHILE->while
PAR IZQ->(
ID->num1
MAYOR IGUAL->>=
NUM->10
PAR DER->)
LLAVE IZQ->{
ID->num2
ASIG->=
ID->num1
DIV->/
NUM->2
PUNTO Y COMA->;
IF->if
PAR IZQ->(
ID->num1
MAYOR->>
NUM->10
PAR DER->)
ID->num1
ASIG->=
```

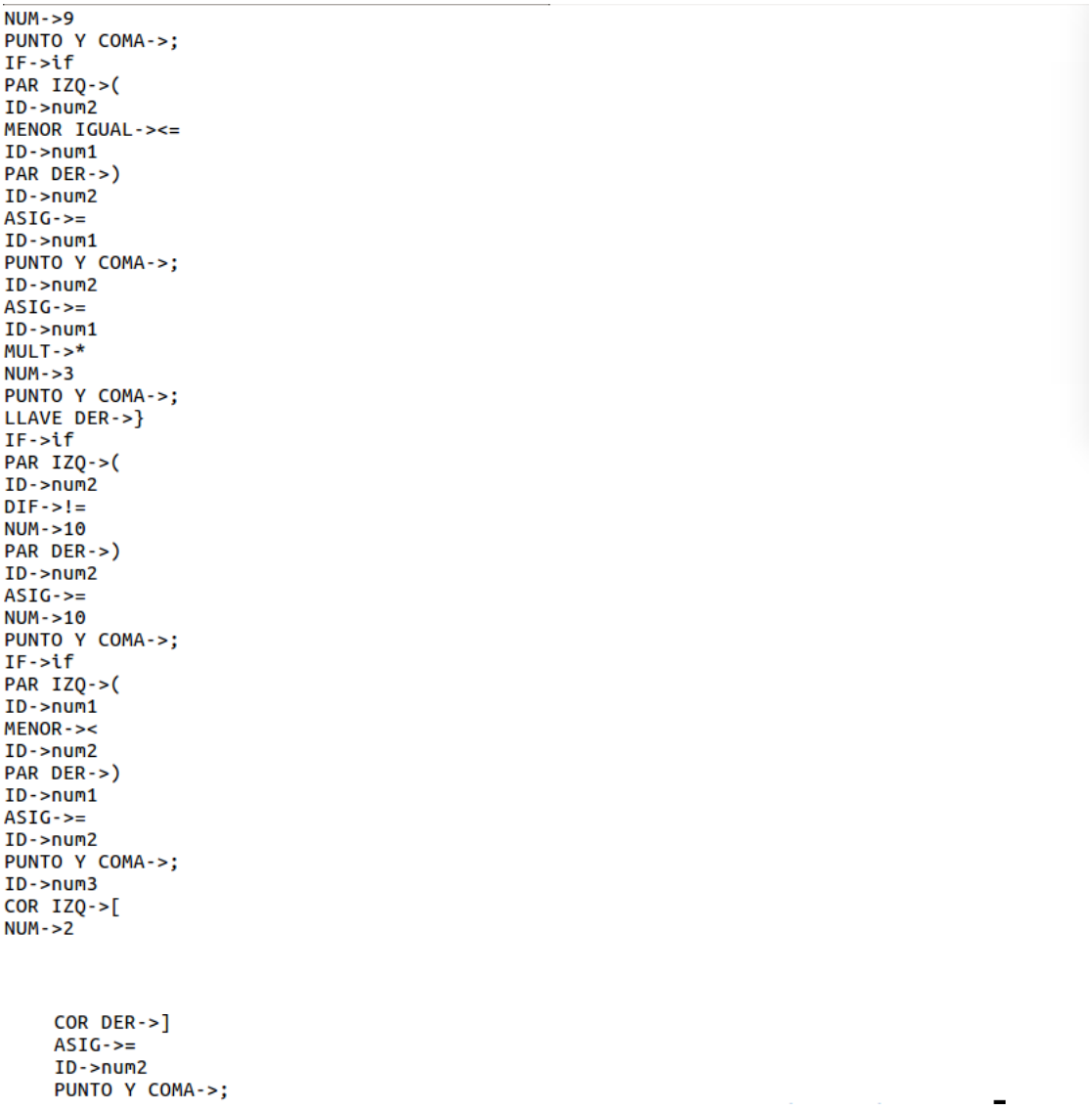


Figura 2: Resultados