

# Analizador Léxico

---

Ariana Bermúdez, Ximena Bolaños, Dylan Rodríguez

Instituto Tecnológico de Costa Rica

---

April 2, 2017

## Análisis Léxico

Se hizo un analizador léxico con la ayuda de la herramienta Flex, para el lenguaje C y que corre en C, este analizador encuentra los tokens y busca su tipo, e incrementa el contador de ese tipo para luego generar histogramas y gráficos de pastel. Estos gráficos son mostrados en una presentación de beamer, que será también la salida del Scanner. En la presentación se definieron los errores con un subrayado rojo, y para los token se definió:

- magenta para los KEYWORD
- rojo para los INTEGER
- verde para los IDENTIFIER
- morado para los CONSTANT
- azul para los OPERATOR
- naranja para los PUNCTUATOR
- rosado para los LITERAL

# FLEX

Herramienta utilizada para realizar escáneres. Este toma los valores de entrada y genera los tokens correspondientes. Según la necesidad del programador.

Este genera un código fuente en C que se va a nombrar `lex.yy.c` en el cual se genera una función `yylex()` la cuál se encarga de analizar el código fuente. Busca la librería `lfl` despues de ser compilado y se enlaza con ella, para dar como resultado un ejecutable.

El fichero de entrada de flex tiene 3 secciones, y tiene que verse como:

definiciones

%%

reglas

%%

código de usuario

Las definiciones contienen las declaraciones de nombres, y condiciones de arranque. Un ejemplo de nuestro programa es:

```
[a-zA-Z][_a-zA-Z0-9]* return IDENTIFIER;  
[0-9][0-9]* return INTEGER;
```

Luego de definir estos campos se procede a explicar como funciona flex, el flex asocia las entradas, **¿pero cómo lo hace?**

El escáner analiza poco a poco las cadenas hasta que concuerden con algún patrón propuesto por el programador. Si se puede emparejar más de una forma entonces tiene prioridad quien pueda asociar más texto y si en ese caso también son iguales entonces se elige por medio de quien esté antes en el fichero de entrada.

El token tendrá asociado el puntero a caracter global **yytext**, y la longitud en la variable global entera **yylen**. Si no hay forma de asociarlo, se usa la regla por defecto.

## FLEX - continuación

### Acciones

Todo patrón tiene una acción asociada. Existen varias acciones, por ejemplo: si se pone

Table: Acciones

%% "texto"	Hará que se borren todas sus apariciones en la entrada.
{	Tomará todo eso como parte de la acción hasta que encuentre la llave que lo cierra, }
-	Hace que la acción actual aplique también para la siguiente.

## FLEX - continuación

### Acciones

El `yylex()` es una función que procesa tokens desde donde lo dejaron la última vez. Directivas especiales que se pueden incluir dentro de una acción

Las acciones puede modificar los `yytext` exepcto su largo, para ello se le puede agregar un `%array` y así modificar totalmente la variable `yytext`, que es donde se guarda el valor del token actual. Directivas especiales:

- **ECHO**: copia `yytext` a la salida del escáner
- **BEGIN**: pone al escáner en la codición de arranque correspondiente.
- **REJECT**: ordena a proceder con al "segunda mejor regla."
- **yymore()**: despues de emparejar una regla el valor de `yytext` actual debe se reemplazado por el siguiente.

# Código Analizado

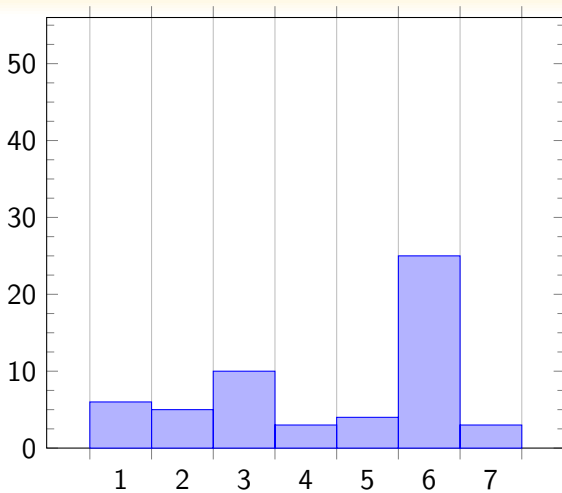
```
holaFuncion ( ) {  
int a = 10 + 3 ;  
int b = 10 + 3 ;  
printf ( " Hello, its me" ) ;  
return true ;  
}  
int funcion ( ) {  
printf ( " Esto es una funcin" ) ;  
return 0 ;
```

# Código Analizado

```
}  
//hola  
//adios oidhofhofaushoi @@ ##  
int funcionPruebaSegunda ( ) {  
printf ( "Prueba" );  
/* int t = c; t++; */  
}
```



# Histograma



## Histograma tipo Pastel

