

Presentacion Beamer

Proyecto 1

Compiladores e Interpretes
Instituto Tecnológico de Costa Rica

2020

Profesor: Dr. Jose Francisco Torres

Examiners:

Nakisha	Dixon
2017103353	
Andrei Leon	201015265
Francisco	Pereira
2017238806	

Tabla de contenidos

Scanner

FLEX (generador de analizador léxico rápido) es una herramienta para generar analizadores léxicos (escáneres o lexers) escrito por Vern Paxson en C alrededor de 1987. Se usa junto con el generador de parser Berkeley Yacc o el generador de parser GNU Bison

Proceso de Scanning

- Flex genera un archivo .C como salida, 'lex.yy.c' (contiene un montón de código incomprensible pero que básicamente contiene las reglas que especificamos en el incluyendo el código para las acciones que especifiquemos). Este archivo contiene una extern function llamada yylex() que escanea un token.

Proceso de Scanning

- Flex genera un archivo .C como salida, 'lex.yy.c' (contiene un montón de código incomprensible pero que básicamente contiene las reglas que especificamos en el incluyendo el código para las acciones que especifiquemos). Este archivo contiene una extern function llamada yylex() que escanea un token.
- Este archivo se compila y se conecta con la librería 'lfl' para producir un ejecutable. Cuando se corre el ejecutable, analiza cierta entrada buscando ocurrencias de las expresiones regulares.

Proceso de Scanning

- Flex genera un archivo .C como salida, 'lex.yy.c' (contiene un montón de código incomprensible pero que básicamente contiene las reglas que especificamos en el incluyendo el código para las acciones que especifiquemos). Este archivo contiene una extern function llamada yylex() que escanea un token.
- Este archivo se compila y se conecta con la librería 'lfl' para producir un ejecutable. Cuando se corre el ejecutable, analiza cierta entrada buscando ocurrencias de las expresiones regulares.
- Cuando encuentra cierta expresión regular, ejecuta cierto código correspondiente que se le asignó.

Codigo fuente

A continuación se presenta el código fuente con colores demostrando la división de *tokens*.

Análisis sintáctico por colores I

```
/* Hello  
  sdfsd  
  World program */
```

```
// Otro comentario
```

```
#include <stdio.h>
```

```
/* here another comment */
```


Análisis sintáctico por colores II

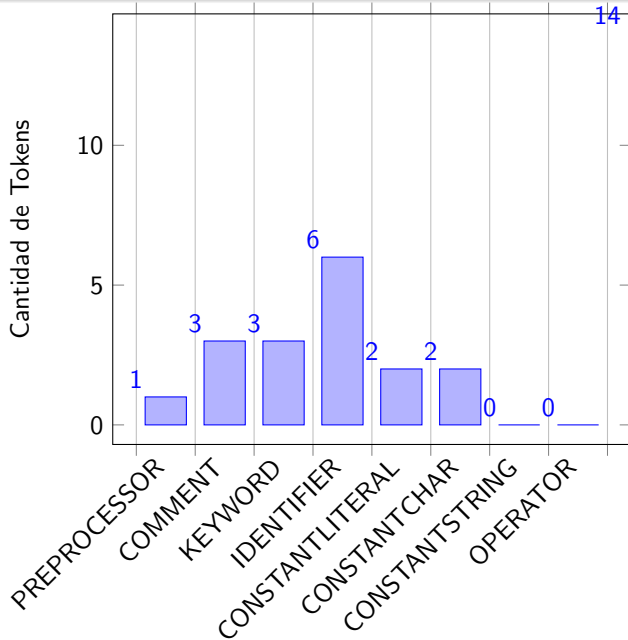
```
main()
{
    int i = i+1;
    signed int i = +1.09982;

    printf("Hellow World");
    printf("Hello World");
}
```

Histograma

A continuación se presenta un histograma que muestra la cantidad de cada tipo de *token* encontrado en el código fuente:

Cantidad de Tokens



Pie Chart

A continuación se presenta un grafico de pie que muestra la cantidad de cada tipo de *token* encontrado en el código fuente: