Presentacion Beamer

Proyecto 1

Compiladores e Interpretes
Instituto Tecnologico de Costa Rica

2020

Profesor: Dr. Jose Francisco Torres

Examiners:

Nakisha Dixon
2017103353
Andrei Leon 201015265
Francisco Pereira
2017238806

Tabla de contenidos

Flex I

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.

Analisis Léxico I

Codigo fuente

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

Análisis sintáctico por colores I

```
#include "test.c"
void main()
{
xprint();
getch();
}
```

Histograma I

Histograma

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

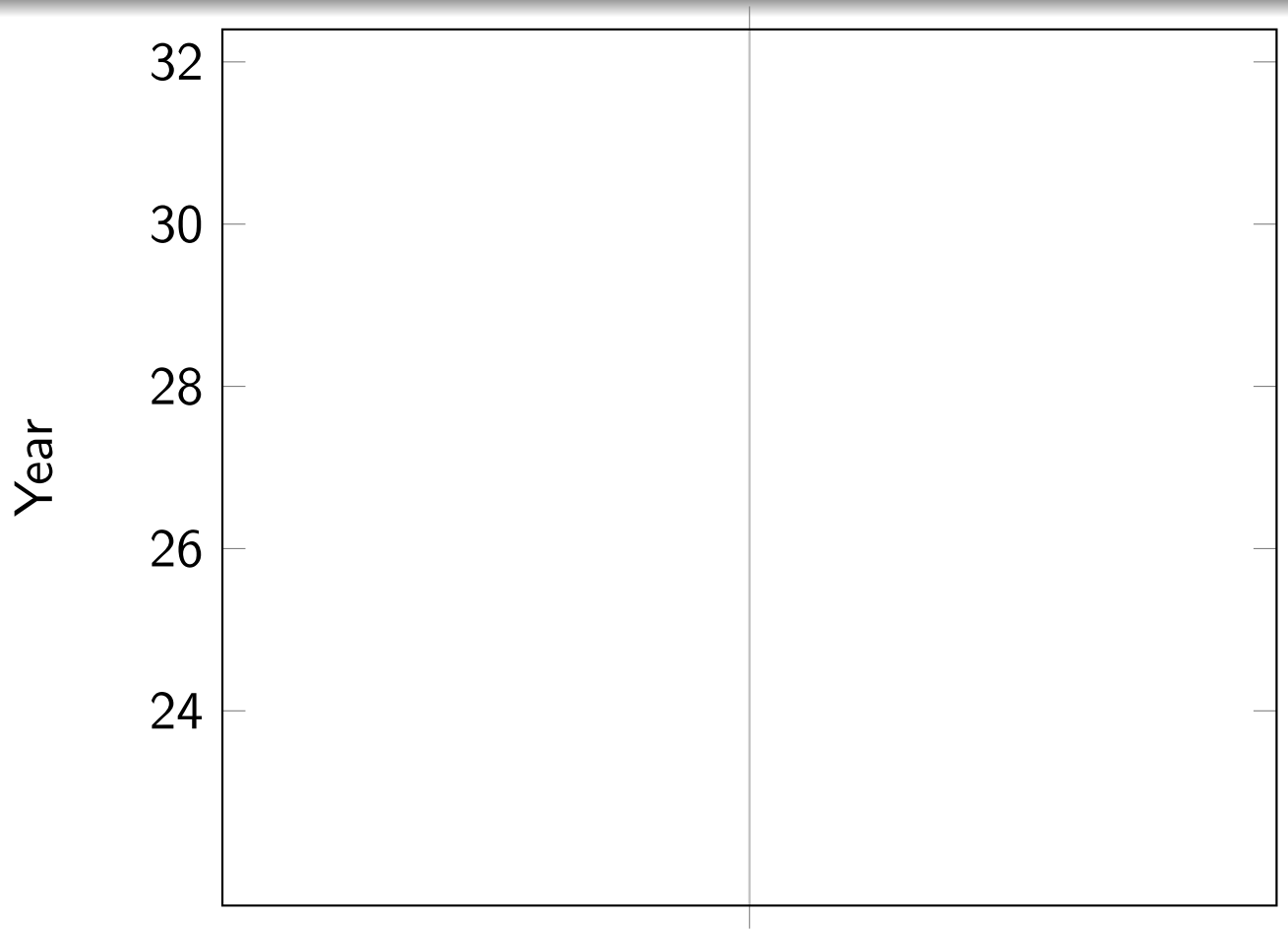
Year
32
30
28
26
24

Grafico Pie I

Pie Chart

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