PROCESADORES DE LENGUAJE

Ingeniería Informática
Especialidad de computación
Tercer curso, segundo cuatrimestre

Departamento de Informática y Análisis Numérico
Escuela Politécnica Superior de Córdoba
Universidad de Córdoba

GUION DE LAS CLASES DE PRÁCTICAS

FLEX/LEX

Contenido

Primer ejemplo mínimo

- **Fichero**
  - minimo.l

- **Descripción**
  - Ejemplo mínimo de flex.
  - El analizador léxico reproduce literalmente todo lo que se teclea.

- **Ejemplo**

  $ ./minimo.exe
  Prueba
  Prueba
  123
  123

Segundo ejemplo mínimo

- **Fichero**
  - ejemplo0.l

- **Descripción**
  - Ejemplo mínimo de flex pero con comentarios
  - El analizador léxico reproduce literalmente todo lo que se teclea

- **Ejemplo**

$ ./ejemplo0.exe
Prueba
Prueba
123
123

Conversión de letras mayúsculas a minúsculas

- **Descripción**
  - El analizador léxico convierte las letras mayúsculas en minúsculas, elimina blancos al final de la línea y sustituye una serie de blancos por uno solo.

- **Fichero**
  - ejemplo1.l

- **Ejemplo**

  $ ./ejemplo1.exe
  Conversión de letras MAYÚSCULAS.
  conversión de letras mayúsculas.

Verbos y no verbos

- **Fichero**
  - ejemplo2.l

- **Descripción**
  - El analizador léxico distingue entre verbos y no verbos

- **Ejemplo**

$ ./ejemplo2.exe
Esta persona es escritor.
Esta: no es un verbo
persona: no es un verbo
es: es un verbo
escritor: no es un verbo
Él puede ser astronauta.
Él: no es un verbo
puede: es un verbo
ser: es un verbo
astronauta: no es un verbo

Verbos, adjetivos, etc.

- **Fichero**
  - ejemplo3.l

- **Descripción**
  - El analizador léxico distingue entre verbos, adjetivos, adverbios, artículos,...

- **Ejemplo de uso**

  $ ./ejemplo3.exe
  él puede ser muy inteligente
  él: es un pronombre
  puede: es un verbo
  ser: es un verbo
  muy: es un adverbio
  inteligente: es un adjetivo

  ella es muy inteligente
  ella: es un pronombre
  es: es un verbo
  muy: es un adverbio
  inteligente: es un adjetivo

  - Observación: el análisis léxico no tiene en cuenta la reglas gramaticales, como muestra el siguiente ejemplo
    muy es inteligente ella
    muy: es un adverbio
    es: es un verbo
    inteligente: es un adjetivo
    ella: es un pronombre

Reconocimiento y almacenamiento de palabras

- **Fichero**
  - ejemplo4.l

- **Descripción**
  - El analizador léxico reconoce las palabras y las almacena en una tabla.
  - Permite definir el tipo de cada palabra.
  - Muestra el uso del operador ^
  - Por ejemplo, si se teclea “al principio de la línea”

    verbo amar luchar

    define las palabras "amar" y "luchar" como verbos

Ejemplos

$ ./ejemplo4.exe
amar luchar
amar: no reconocida
luchar: no reconocida

verbo amar luchar
amar luchar
amar: verbo
luchar: verbo

verbo jugar
verbo: no reconocida
jugar: no reconocida
jugar
jugar: no reconocida
verbo jugar
jugar
jugar: verbo

ella es muy inteligente
ella: no reconocida
es: no reconocida
muy: no reconocida
inteligente: no reconocida

adj inteligente
verbo es
pron ella
adv muy

ella es muy inteligente
ella: pronombre
es: verbo
muy: adverbio
inteligente: adjetivo

pron inteligente
---> : la palabra inteligente ya está definida

Reconocimiento y almacenamiento de palabras, segunda versión

- **Fichero**
  - ejemplo44.l
  - p44.c

- **Descripción**
  - Realiza las mismas acciones que el ejemplo 4, pero utiliza un fichero auxiliar del lenguaje C para separar el análisis léxico de la gestión de la lista de palabras.

Componentes léxicos de una lenguaje de programación, primera versión

- **Ficheros**
  - ejemplo5.l
  - ejemplo5.h

- **Descripción**
  - El analizador léxico reconoce algunos componentes léxicos de un lenguaje de programación.
  - Finaliza el programa cuando se teclea el carácter # al principio de la línea.

- **Ejemplo**

```
$ ./ejemplo5.exe
if (a>0) then b = a; else b = -a;
 Palabra reservada: if --> token 257
(
 Identificador: a --> token 260
 Operador relacional: > --> token 262
 Numero: 0 --> token 261
)
 Palabra reservada: then --> token 258
 Identificador: b --> token 260
=
 Identificador: a --> token 260
;
 Palabra reservada: else --> token 259
 Identificador: b --> token 260
=-
 Identificador: a --> token 260
;
  #
#
#
 Fin del programa
```

Componentes léxicos de una lenguaje de programación, segunda versión

- **Ficheros**
  - ejemplo55.l
  - ejemplo5.h
  - entrada.txt

- **Descripción**
  - Permite el uso de argumentos desde la línea de comandos para indicar el fichero que se desea analizar.

- **Ejemplo**

```
$ ./ejemplo55.exe entrada.txt
 Palabra reservada: if --> token 257
(
 Identificador: dato --> token 260
 Operador relacional: > --> token 262
 Numero: 0 --> token 261
)
 Identificador: valor --> token 260
=
 Identificador: dato --> token 260
;
 Palabra reservada: else --> token 259
 Identificador: valor --> token 260
=-
 Identificador: dato --> token 260
```

Comando REJECT

- **Fichero**
  - pink.l

- **Descripción**
  - Muestra el uso de REJECT
  - Permite que un texto de entrada pueda ser asociado a más de una expresión regular.
  - Después de emparejar el texto de entrada con una expresión regular, la rechaza para comprobar si el texto puede asociarse a otra expresión regular

- **Ejemplo**

  $ ./pink.exe
  pint
  pink
  int
  Crtrl^D
  Contador de palabras
  pink = 1
  ink = 1
  pin = 2

Función yymore()

- **Fichero**
  - hiper.l

- **Descripción**
  - Muestra el uso de *yymore()*
  - Permite concatenar el siguiente texto que sea reconocido con el contenido actual de *yytext*.

- **Ejemplo**

$ ./hiper.exe
texto
Token = texto
hipertexto
Token = hipertexto
mercado
Token = mercado
hipermercado
Token = hipermercado
enlace
hiperenlace

Reconocimiento de comentarios

- **Fichero**
  - comentario.l
- **Descripción**
  - El analizador léxico reconoce y cuenta los comentarios del lenguaje C
  - Muestra el comentario reconocido y el número de líneas.
  - Muestra el uso de estados de Flex
    - Comando BEGIN
    - Reglas condicionales controladas por un estado de flex definido por el programador
- **Ejemplo**

```
$ ./comentario.exe

/* Comentario de una línea */
 nº comentario = 1, lineasComentario = 1
/*
Comentario
de
varias
líneas
*/
 nº comentario = 2, lineasComentario = 6
/*
Comentario anidado
 /*
 No se pueden anidar comentarios
```

AFD-Comentarios1.l

- **Fichero**
  - AFD-Comentarios1.l
- **Descripción**
  - El analizador léxico reconoce y cuenta los comentarios del lenguaje C
  - No muestra el comentario reconocido ni el número de líneas.
  - Muestra el uso de estados de Flex
    - Comando BEGIN
    - Reglas condicionales controladas por estados de flex definidos por el programador
- **Ejemplo**

$ ./AFD-Comentarios1.exe

/ Comentario de una línea */*

Comentario reconocido

/
Comentario
de
varias
líneas
**/*
Comentario reconocido

/
Comentario anidado
/
Error: comentario anidado

AFD-Comentarios2.l

- **Fichero**
  - AFD-Comentarios2.l
- **Descripción**
  - El analizador léxico reconoce y cuenta los comentarios del lenguaje C
  - Muestra el comentario reconocido.
  - Muestra el uso de estados de Flex
    - Comando BEGIN
    - Reglas condicionales controladas por un estado de flex definido por el programador
    - Función yymore()
- **Ejemplo**

$ ./AFD-Comentarios2.exe

/ Comentario de una línea */*

Comentario reconocido: / Comentario de una línea */*

/
Comentario
de
varias
líneas
/
Comentario reconocido: /
Comentario
de
varias
líneas
/

/
Comentario anidado
/
Error: comentario anidado

Reconocimiento de cadenas, primera versión

- **Fichero**
  - cadena_1.l

- **Descripción**
  - Reconoce cadenas delimitadas por comillas simples

- **Ejemplo**

$ ./cadena_1.exe
'Ejemplo de cadena'
Cadena reconocida = 'Ejemplo de cadena'

'Cadena
escrita
en varias líneas'
Cadena reconocida = 'Cadena
escrita
en varias líneas'

'Cadena con \'comillas\' internas'
Cadena reconocida = 'Cadena con \'comillas\' internas'

Reconocimiento de cadenas, segunda versión

- **Fichero**
  - cadena_2.l

- **Descripción**
  - Reconoce y cuenta cadenas delimitadas por comillas simples
  - Numera las cadenas e indica cuántas líneas contiene.
  - Se muestra el uso de
    - Comando BEGIN
    - Reglas condicionales controladas por un estado de flex definido por el programador
    - Función yymore()

- **Ejemplo**

```
$ ./cadena_2.exe
'Ejemplo de cadena'

nº cadena = 1, lineas_cadenas = 1
        Cadena reconocida = Ejemplo de cadena

'Cadena escrita
en
varias líneas'

nº cadena = 2, lineas_cadenas = 3
        Cadena reconocida = Cadena escrita
en
varias líneas

'Cadena con \'comillas\' internas'

nº cadena = 3, lineas_cadenas = 1
        Cadena reconocida = Cadena con \'comillas\' internas
```

AFD-Cadenas1

- **Fichero**
  - AFD-Cadenas_1.l

- **Descripción**
  - Reconoce cadenas delimitadas por comillas simples
  - No muestra la cadena reconocida
  - Se simula el funcionamiento de un autómata finito determinista (AFD)
  - Se muestra el uso de
    - Comando BEGIN
    - Reglas condicionales controladas por estados de flex definidos por el programador

- **Ejemplo**

$ ./AFD-Cadenas_1.exe
'Ejemplo de cadena'
Cadena reconocida

'Cadena
escrita
en varias líneas'
Cadena reconocida

'Cadena con \'comillas\' internas'
Cadena reconocida

AFD-Cadenas2

- **Fichero**
  - AFD-Cadenas_2.l

- **Descripción**
  - Reconoce y cuenta cadenas delimitadas por comillas simples
  - Muestra la cadena reconocida
  - Se simula el funcionamiento de un autómata finito determinista (AFD)
  - Se muestra el uso de
    - Comando BEGIN
    - Reglas condicionales controladas por estados de flex definidos por el programador
    - Función yymore()

- **Ejemplo**

```
$ ./AFD-Cadenas_1.exe
'Ejemplo de cadena'

nº cadena = 1, lineas_cadenas = 1
        Cadena reconocida = Ejemplo de cadena

'Cadena escrita
en
varias líneas'

nº cadena = 2, lineas_cadenas = 3
        Cadena reconocida = Cadena escrita
en
varias líneas

'Cadena con \'comillas\' internas'

nº cadena = 3, lineas_cadenas = 1
        Cadena reconocida = Cadena con \'comillas\' internas
```