Analizador Léxico-Sintáctico

Compiladores

Hernández Gómez Ricardo

Profesora: Sandoval Montaño Laura

Grupo: 2

Fecha: 30/10/2018

Descripción

Analizador sintáctico hecho en lenguaje C para la gramática Pu+ elaborada durante la clase de Compiladores.

Partiendo del análisis léxico previamente realizado con Flex, se implementaron las funciones correspondientes para poder generar una cadena de átomos a partir de los componentes léxico detectados. Esta cadena es esencial para el análisis sintáctico.

Son relevantes las clases de componentes léxicos. Las clases con más variantes son las de palabras reservadas y operadores relacionales.

Las reglas de la gramática son las siguientes:

1:	G → [Z]	**27:**	K → s
2:	Z → DZ	**28:**	K → E
3:	Z → Ɛ	**29:**	R → EQ
4:	Z → Y	**30:**	Q → OE
5:	Y → SX	**31:**	Q → Ɛ
6:	X → Y	**32:**	O → !
7:	X → Ɛ	**33:**	O → q
8:	D → JaV	**34:**	O → <
9:	J → b	**35:**	O → l
10:	J → c	**36:**	O → >
11:	J → e	**37:**	O → g
12:	J → d	**38:**	E → TE’
13:	V → ,aV	**39:**	E’ → +TE’
14:	V → ;	**40:**	E’ → -TE’
15:	S → A;	**41:**	E’ → Ɛ
16:	S → H	**42:**	T → FT’
17:	S → M	**43:**	T’ → *FT’
18:	S → P	**44:**	T’ → /FT’
19:	S → I	**45:**	T’ → %FT’
20:	A → a=K	**46:**	T’ → Ɛ
21:	H → h[Y]m(R);	**47:**	F → (E)
22:	M → m(R)[Y]	**48:**	F → a
23:	P → p(A;R;A)[Y]	**49:**	F → n
24:	I → i(R)[Y]N	**50:**	F → r
25:	N → Ɛ	**51:**	K → s
26:	N → o[Y]	**52:**	K → s

Tomando estas reglas como base se realizó la comprobación de gramática LL(1) (el procedimiento se muestra en la siguiente sección). Con esto, es posible definir los conjuntos de

selección para cada producción; se utilizan en la implementación de las funciones recursivas que permiten el reconocimiento sintáctico.

El análisis sintáctico se ejecuta sólo hasta que se ha realizado el análisis léxico, pues es necesaria como entrada la cadena de átomos generada; una vez finalizado se regresa la cadena de átomos, así como un mensaje indicando si la sintaxis es correcta o existen errores.

Átomos para cada clase:

- *Palabras reservadas:*
  - Bul: b
  - Cadena: c
  - Cierto: t
  - Falso: f
  - Haz: h
  - Mientras: m
  - Para: p
  - Real: d
  - Si: i
  - Sino: o
- *Identificadores:*
  - a
- *Símbolos especiales:*
  - ( ) , ; [ ]
- *Operador de asignación:*
  - =
- *Operadores relacionales:*
  - .DIF. : i
  - .IGL. : q
  - .MN. : <
  - .MNI. : l
  - .MY. : >
  - .MYI. : g
- *Operadores aritméticos:*
  - + - * / %
- *Constante cadena:*
  - s
- *Constantes numéricas enteras:*
  - n
- *Constantes numéricas reales:*
  - r

Análisis

- *Planificación:*

Actividad	Elabora
Comprobación LL(1)	Ricardo Hernández Gómez
Implementación de función generadora de la cadena de átomos en C	Ricardo Hernández Gómez
Implementación de las funciones de análisis en C	Ricardo Hernández Gómez
Pruebas	Ricardo Hernández Gómez

- *Diseño:*
  - Arreglos de átomos de palabras reservadas y átomos de operadores relacionales:

Los átomos de las palabras reservadas y los operadores relacionales están definidos por defecto en archivos .txt ubicados en el mismo directorio que el programa. Los átomos están definidos con el siguiente formato.

Átomo1

Átomo2

Átomo3

.

.

.

La función "populateArray(char *fileName, int *numero)" se ejecuta para ambos, insertándose en sus respectivos arreglos conforme se va leyendo, línea por línea. Debido a esto es posible omitir el número de su valor del archivo, pues la posición en la que son insertados en el arreglo de palabras reservadas es este. Las posiciones corresponden a las de los arreglos de palabras reservadas y operadores relacionales, respectivamente.

  - Cálculo de conjuntos de selección y comprobación LL(1):

a) Producciones anulables: 3, 7, 25, 31, 41, 46
   No-terminales anulables: Z, X, N, Q, E', T'
b) Firsts:

First	1	=	[
First	2	=	b c e d
First	3	=	
First	4	=	a h m p i

First	5	=	a h m p i
First	6	=	a h m p i
First	7	=	
First	8	=	b c e d

First	9	=	b
First	10	=	c
First	11	=	e
First	12	=	d
First	13	=	,
First	14	=	;
First	15	=	a
First	16	=	h
First	17	=	m
First	18	=	p
First	19	=	i
First	20	=	a
First	21	=	h
First	22	=	m
First	23	=	p
First	24	=	i
First	25	=	
First	26	=	o
First	27	=	s
First	28	=	(a n r
First	29	=	(a n r
First	30	=	> < q l g !

First	31	=	
First	32	=	!
First	33	=	q
First	34	=	<
First	35	=	l
First	36	=	>
First	37	=	g
First	38	=	(a n r
First	39	=	+
First	40	=	-
First	41	=	
First	42	=	(a n r
First	43	=	*
First	44	=	/
First	45	=	%
First	46	=	
First	47	=	(
First	48	=	a
First	49	=	n
First	50	=	r
First	51	=	t
First	52	=	f

Para cada no terminal:

First(G) =	[
First(Z) =	b c e d a h m p i
First(Y) =	a h m p i
First(X) =	a h m p i
First(D) =	b c e d
First(J) =	b c e d
First(V) =	, ;
First(S) =	a h m p i
First(A) =	a
First(H) =	h
First(M) =	m
First(P) =	p

First(I) =	i
First(N) =	o
First(K) =	s (a n r t f
First® =	(a n r
First(Q) =	> < q l g !
First(O) =	> < q l g !
First€ =	(a n r
First(E') =	+ -
First(T) =	(a n r
First(T') =	* / %
First(F) =	(a n r

c) Follows:

Follow(G) =		

Follow(Z) =	] fin	] fin
Follow(Y) =	Follow(Z) U {fin]}	] fin
Follow(X) =	Follow(Y) U {fin}	] fin
Follow(D) =	First(Z) U Follow(Z)	b c e d a h m p i] fin
Follow(J) =	a	a
Follow(V) =	Follow(D) U fin	b c e d a h m p i] fin
Follow(S) =	First(X) U Follow(Y)	a h m p i] fin
Follow(A) =	;)	;)
Follow(H) =	Follow(S) U fin	a h m p i] fin
Follow(M) =	Follow(S) U fin	a h m p i] fin
Follow(P) =	Follow(S) U fin	a h m p i] fin
Follow(I) =	Follow(S) U fin	a h m p i] fin
Follow(N) =	Follow(I) U fin	a h m p i] fin
Follow(K) =	Follow(A) U fin	;) fin
Follow(R) =	) ;	) ;
Follow(Q) =	Follow(R U fin	) ; fin
Follow(O) =	First(E	(a n r
Follow(E) =	First(Q) U Follow(R U Follow(Q) U) U Follow(K)	> < q l g !) ; fin
Follow(E') =	Follow(E U fin	> < q l g !) ; fin
Follow(T) =	First(E') U Follow(E')	> < q l g !) ; fin + -
Follow(T') =	Follow(T) U fin U	> < q l g !) ; fin + -
Follow(F) =	First(T') U Follow(T') U Follow(T)	> < q l g !) ; fin + - * / %

d) Conjuntos de selección (C.S.):

C.S.	1	=	[
C.S.	2	=	b c e d
C.S.	3	=	] fin
C.S.	4	=	a h m p i
C.S.	5	=	a h m p i
C.S.	6	=	a h m p i
C.S.	7	=	] fin
C.S.	8	=	b c e d
C.S.	9	=	b
C.S.	10	=	c
C.S.	11	=	e
C.S.	12	=	d
C.S.	13	=	,
C.S.	14	=	;
C.S.	15	=	a
C.S.	16	=	h

C.S.	17	=	m
C.S.	18	=	p
C.S.	19	=	i
C.S.	20	=	a
C.S.	21	=	h
C.S.	22	=	m
C.S.	23	=	p
C.S.	24	=	i
C.S.	25	=	a h m p i] fin
C.S.	26	=	o
C.S.	27	=	s
C.S.	28	=	(a n r
C.S.	29	=	(a n r
C.S.	30	=	> < q l g !
C.S.	31	=	) ; fin
C.S.	32	=	!

C.S.	33	=	q
C.S.	34	=	<
C.S.	35	=	l
C.S.	36	=	>
C.S.	37	=	g
C.S.	38	=	(a n r
C.S.	39	=	+
C.S.	40	=	-
C.S.	41	=	> < q l g !) ; fin
C.S.	42	=	(a n r

C.S.	43	=	*
C.S.	44	=	/
C.S.	45	=	%
C.S.	46	=	> < q l g !) ; fin + -
C.S.	47	=	(
C.S.	48	=	a
C.S.	49	=	n
C.S.	50	=	r
C.S.	51	=	t
C.S.	52	=	F

Todos los conjuntos de selección son disjuntos, por lo que se trata de una gramática LL(1). Se puede proceder a la implementación en C.

- *Implementación:*

La implementación se realizará en lenguaje C, partiendo del analizador Léxico realizado previamente con la ayuda de Flex.

La implementación se realizó utilizando la distribución de Linux, Ubuntu, y la compilación se realizó utilizando gcc.

Instrucciones de ejecución

Para para poder ejecutar el programa es necesario generar el código en C y compilarlo desde Linux utilizando el compilador GCC.

Comenzamos por generar el código en archivo .C utilizando Flex:

```
$ flex analizador.l
```

Esto generará el archivo .C con el nombre “lex.yy.c”. Procedemos a compilarlo:

```
$ gcc lex.yy.c -lfl -o analizador
```

Una vez hecho esto podemos ejecutar el programa. Para su uso es necesario indicar el nombre del archivo a analizar inmediatamente después del programa.

```
$ ./analizador testfile2.txt
```

El programa se ejecutará e imprimirá las tablas en pantalla, además de guardarlas como archivo de texto en la misma carpeta para su revisión.

Se muestra la cadena de átomos y si se trata de una sintaxis correcta o no.

Conclusiones

El análisis sintáctico es el segundo tipo de análisis que se realiza en un proceso de compilación. Se trata de un proceso indispensable, pues es gracias a este que se puede asegurar, o no, que el orden de los componentes léxicos es el correcto, de acuerdo con las reglas de producción de la gramática establecida.

Gracias al trabajo realizado para la primera etapa de compilación, el análisis léxico, fue posible realizar una implementación de este proceso de manera apropiada, realizando algunas modificaciones e implementando la funcionalidad en lenguaje C.

Con este proyecto se pudo comprobar y aplicar lo visto en clase con respecto al tema, pudiendo apreciar claramente la necesidad de los procesos descritos en este documento, así como aprender las particularidades de cada uno.

Ya finalizado el proceso de pruebas, puedo decir que el objetivo se cumplió, realizándose exitosamente la comprobación de cada regla de producción gramatical previamente definida.