Compilador lexicográfico básico
“El Automatador”

Practica 2:
Gramáticas de reconocimiento descendente

Instructor
Roberto Flórez Rueda

Estudiante
Roy Maestre - 1214715506

UdeA
Facultad de Ingeniería
Departamento de Ingeniería de Sistemas
Teoría de Lenguajes y Laboratorio

2020-1

Gramática de Reconocimiento.

Para la realización de esta segunda practica se construyó una gramática Q que permitiera reconocer código del lenguaje de programación Java. La gramática se basó en el autómata finito llamado “autómata léxico” presentado en la primera práctica. Con algunos ajustes extra como permitir el reconocimiento y diferenciación de paréntesis que abren y cierra, al igual que las llaves.

La gramática construida se presenta a continuación, además de los grupos de selección correspondientes a cada producción.

Para facilitar la lectura de la gramática se presentan los diferentes terminales.

1. <S>: Estado inicial
2. <T>: Estado cuando se ingresa un tipo de variable (Ej: String, int…)
3. <V>: Estado cuando se ingresa una variable
4. <TV>: Estado cuando se ingresa un tipo seguido de una variable.
5. <V=>: Estado cuando se ingresa una variable y un símbolo de asignación o modificación de valor. (Ej: casa =, x +=…)
6. <TV,>: Estado cuando se ingresa un tipo, una variable y luego una coma.
7. <TV=>: Estado para cuando se ingresa un tipo, una variable y un símbolo de asignación.
8. <V=V>: Estado para cuando se ingresa una variable, un símbolo de asignación y otra variable.
9. <V=d>: Estado para cuando se ingresa una variable, un símbolo de asignación y una constante numérica o cadena de texto.
10. <TV=V>: Estado para cuando se ingresa un tipo, una variable, un símbolo de asignación y una variable.
11. <TV=d>: Estado para cuando se ingresa un tipo, una variable, un símbolo de asignación y unas constante o cadena de caracteres.
12. <V=VOp>: Estado para cuando se ingresa una variable, un símbolo de asignación, otra variable y un operador aritmético o lógico.
13. <TV=VOp>: Estado para cuando se ingresa un tipo, una variable, un símbolo de asignación, otra variable y un operador aritmético o lógico.
14. <Expresion>: Estado para identificar expresiones booleanas en los paréntesis de una secuencia de decisión o iterativa.
15. <RestoExpresion: Estado auxiliar para expresiones booleanas en paréntesis.

Producción	Conjunto Selección
1. <S> → Tipo <T> FinDeLinea<S>	1. Tipo
2. <S> → Variable <V> FinDeLinea<S>	2. Variable
3. <S> → Decision (<expresion>){<S>} <Else>FinDeLinea<S>	3. Decision
4. <S> → λ	4. fin de secuencia, }
5. <T> → Variable <TV>	5. Variable
6. <V> → Asignación <V=>	6. Asignación
7. <V> → Modificador <V+=>	7. Modificador
8. <V> → Suma/Resta 1	8. Suma/Resta 1
9. <TV> → Separador <TV,>	9. Separador
10. <TV> → Asignación <TV=>	10. Asignación
11. <TV> → λ	11.), fin de linea
12. <V=> → Variable <V=V>	12. Variable
13. <V=> → Int <V=d>	13. Int
14. <V=> → Float <V=d>	14. Float

15. <V=> → Double <V=d>	15. Double
16. <V=> → Bool <V=d>	16. Bool
17. <V=> → String <V=d>	17. String
18. <V=> → “<Cadena>”<V=d>	18. "
19. <V=> → (<V=>)	19. (
20. <TV,> → Variable <TV>	20. Variable
21. <TV=> → Variable <TV=V>	21. Variable
22. <TV=> → Int <TV=d>	22. Int
23. <TV=> → Float <TV=d>	23. Float
24. <TV=> → Double <TV=d>	24. Double
25. <TV=> → Bool <TV=d>	25. Bool
26. <TV=> → String <TV=d>	26. String
27. <TV=> → “<Cadena>”<TV=d>	27. "
28. <TV=> → (<TV=>)	28. (
29. <V=V> → OAritmetico<V=VOp>	29. Oaritmetico
30. <V=V> → OBooleano <V=VOp>	30. OBooleano
31. <V=V> → Suma/Resta 1	31. Suma/Resta 1
32. <V=V> → λ	32.), fin de linea
33. <V=d> → OAritmetico<V=VOp>	33. Oaritmetico
34. <V=d> → OBooleano <V=VOp>	34. OBooleano
35. <V=d> → λ	35.), fin de linea
36. <TV=V> →Separador<TV,>	36. Separador
37. <TV=V> → OAritmetico<TV=VOp>	37. Oaritmetico
38. <TV=V> → OBooleano<TV=VOp>	38. OBooleano
39. <TV=V> → Suma/Resta 1	39. Suma/Resta 1
40. <TV=V> → λ	40.), fin de linea
41. <TV=d> → Separador <TV,>	41. Separador
42. <TV=d> → OAritmetico<TV=VOp>	42. Oaritmetico
43. <TV=d> → OBooleano <TV=VOp>	43. OBooleano
44. <TV=d> → λ	44.), fin de linea
45. <V=VOp> → Variable <V=V>	45. Variable
46. <V=VOp> → Int <V=V>	46. Int
47. <V=VOp> → Float <V=V>	47. Float
48. <V=VOp> → Double <V=V>	48. Double
49. <V=VOp> → Bool <V=V>	49. Bool
50. <V=VOp> → String <V=V>	50. String
51. <V= VOp> → “<Cadena>”<V=V>	51. "
52. <V=VOp> → (<V=VOp>)	52. (
53. <TV=VOp> → Variable <TV=V>	53. Variable
54. <TV=VOp> → Int <TV=V>	54. Int
55. <TV=VOp> → Float <TV=V>	55. Float
56. <TV=VOp> → Double <TV=V>	56. Double
57. <TV=VOp> → Bool <TV=V>	57. Bool
58. <TV=VOp> → String <TV=V>	58. String
59. <TV=VOp> → “<Cadena>”<TV=V>	59. "
60. <TV=VOp> → (<TV=VOp>)	60. (
61. <Expresion> → Variable <Expresion2>	61. Variable
62. <Expresion> → Bool	62. Bool
63. <Expresion> → (<Expresion>)	63. (
64. <Expresion2> → OBooleano <Expresion>	64. OBooleano
65. <Expresion2> → λ	65.)
66. <Else> → else {<S>}	66. else

67. <Else> → λ	67. fin de linea

RESTRUCCIONES DE LA GRAMATICA

La gramática reconocerá:

- Variables booleanas (true, false).
- Cadenas de caracteres y variables tipo string.
- Variables numéricas (int, float, double).
- Sentencias de decision (if, else).
- Sentencias de iteración (while, for). Pero para la sentencia for no se identifica la integridad de la estructura dentro de los paréntesis que la siguen.

Para evitar que el programa paré al encontrar un error, se altero el reconocedor recursivo para que al encontrar un error avance a la siguiente línea. Sin embargo, si el error implica la ausencia de punto y coma al final de la línea, se considera la siguiente línea como la misma, así que se avanzará hasta encontrar un punto y coma.

Para cualquier otra duda sobre los símbolos de la gramática remitirse a el documento “De autómata a código” presentado para la primera practica y que se encuentra en este mismo repositorio de GitHub.

Para conocer el manejo de la aplicación remitirse a la sección final del manual de usuario.

Los documentos de explicación de como abrir y correr el programa de la primera practica son válidos para esta también.

CAMBIOS EN EL CODIGO DE LA APLICACIÓN.

- Se agregó la clase ReconocedorRecursivo en la cual se encuentra la mayor parte del código desarrollado para esta segunda practica.
- Se presentan cambios mínimos en el autómata de palabras reservadas y en el autómata de símbolos para reconocer paréntesis y llaves.
- Se agrega dos métodos a la clase controller, uno que da inicio a el reconocedor recursivo y el segundo que permite hacer un preprocesado al texto inicial.
- El resto del código no presenta cambios.

NOTA FINAL:

Mi compañero de primera practica Juan Cardona canceló la materia, por lo que esta fue realiza individualmente, tener piedad por favor.